(1)

FINAL TECHNICAL REPORT

to

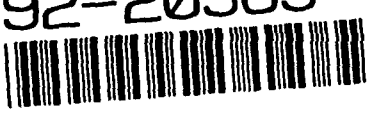
OFFICE OF NAVAL RESEARCH

Contract No.: N00014-86-K-0542

Principal Investigator: N. K. Bose
The Pennsylvania State University

Program Manager: R. N. Madan

July 1992



FINAL TECHNICAL REPORT

submitted to

OFFICE OF NAVAL RESEARCH
(ATTENTION: DR. RABINDER N. MADAN)
1114 SE, Code 414
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Name and Address of Institution	The Pennsylvania State University University Park, PA 16802
Name of Principal Investigator	Dr. N. K. Bose, HRB-Systems Professor Department of Electrical and Computer Engineering
Grant No.	N00014-86-K-0542
Starting Date	6/1/86
Completion Date	5/15/92
Grant Title	Application of Smoothing Techniques for Tracking Maneuvering Targets (6/1/86-2/28/89) Multiple Target Tracking in Clutter: New Approaches (3/1/89-5/15/92)

Statement A per telecon Rabinder Mandan
ONR/Code 1114
Arlington, VA 22217-5000

NWW 7/29/92

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1992	Final June '86 – May '92

4. TITLE AND SUBTITLE Application of Smoothing Techniques for Tracking Maneuvering Targets (6/1/86–2/18/89)
Multiple Target Tracking in Clutter: New Approaches (3/1/89–5/15/92)

5. FUNDING NUMBERS
SFRC Number
N00014–86–K–0542

6. AUTHOR(S)
N. K. Bose

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Pennsylvania State University
University Park, PA 16802

8. PERFORMING ORGANIZATION REPORT NUMBER
Code 7A720

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Resident Representative N66005
Office of Naval Research
The Ohio State University Research Center
1314 Kinnear Road
Columbus, Ohio 43212–1194

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
Code N66005

11. SUPPLEMENTARY NOTES
Scientific Program Officer: Dr. R. N. Madan
Office of Naval Research, Code :1114SE
800 N. Quincy Street, Arlington, VA 22217–5000

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Scientific Program Officer, Administrative Grants Officer, Director, Naval Research Laboratory, Defense Technical Information Center

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The project was concerned with development of fast algorithms for clustering and data association and then applying fixed-lag smoothers for tracking maneuvering and non-maneuvering multitargets in clutter. The data association was done by a depth-first search (DFS) technique which extends naturally to three dimensional validation matrices which occur in Markov models of system parameters for maneuvering targets and also in multiscan correlation. A faster algorithm is then suggested for computing approximately the a posteriori probabilities without generating the data association hypotheses. This algorithm is suitable for implementation with multiprocessors in mobile airborne or seaborne distributed tracking facilities, while the DFS-based approach is suitable for ground-based trackers. Simulation studies have been conducted to demonstrate the advantages of the algorithms developed in this project over comparable ones existing in the literature.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Multitarget Tracking, JPDAF, JPDAS, Data Association, Fast Algorithms for Tracking, Depth-First Search	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

CONTENTS

1. Research Summary

This research tackles the problems of data association and state estimation in the multitarget tracking of non-maneuvering as well as maneuvering targets in clutter. The major contributions are the development of a depth-first search (DFS) algorithm for data association, three algorithms which are suitable for implementation in multiprocessor systems for fast computation of the *a posteriori* probabilities of the origins of measurements in the joint probabilistic data association filter (JPDAF), and a joint probabilistic data association fixed-lag smoother (JPDAS) for improving the accuracy of state estimation.

In the DFS algorithm, the problem of data association is modelled as an exhaustive search problem. Based on this model, a specialized DFS approach is proposed for efficiently generating the data association hypotheses and computing rapidly the conditional probabilities of the hypotheses for data association in the JPDAF. In the three algorithms developed for implementation on multiprocessor systems, the *a posteriori* probability of the origin of each measurement in the JPDAF is decomposed into two parts. The computation of one part becomes trivial and the different ways for computing the other part lead to the development of the three algorithms. The computational costs and memory requirements of the above algorithms are analyzed in the worst case as well as in the average case.

A comprehensive analysis reveals several drawbacks of the neural network probabilistic data association (NPDA) algorithm. The NPDA is an application of the Hopfield neural network to the data association problem.

In order to improve the accuracy of state estimation for multitarget tracking in clutter, an algorithm is developed by making use of the joint probabilistic data association fixed-lag smoothing techniques. It is shown that a significant improvement in the accuracy of state estimation of both nonmaneuvering and maneuvering targets may be achieved by introducing a time lag of one or two sampling periods between the instants of estimation and the latest measurement. Finally, a computer simulation system is developed to demonstrate the effectiveness of the developed algorithms.

2. Research Description

This research is mainly focused on data association and target state estimation in multitarget tracking in a cluttered environment. The objectives are to design algorithms for efficiently generating the data association hypotheses, computing rapidly the *a posteriori* probabilities, β_j^t's, and improving the accuracy of target state estimation. The major contributions of this research are the development of

- the depth-first search (DFS) algorithm for data association,
- three algorithms for computing the *a posteriori* probabilities using multiprocessor systems, and
- the JPDA fixed-lag smoothing (JPDAS) algorithm.

In the DFS algorithm, the problem of data association is identified as an *exhaustive search problem* in general. Subsequently, a mathematical model is proposed for the problem of data association in the JPDAF. Based on the model, a specialized DFS approach is developed for the fast generation of data association hypotheses and for the computation of the conditional probabilities of the hypotheses in the JPDAF. The computational complexity of the algorithm is analyzed in terms of the number of multiplications and additions required in the computation of the *a posteriori* probabilities, β_j^t's, in the worst case as well as in the average case. Although the DFS algorithm requires more multiplications and additions that the fast JPDA algorithm [1] in the worst case, it is much more efficient than the fast JPDA algorithm in the average case. In addition, the DFS algorithm requires less memory than the fast JPDA algorithm. Another advantage of the DFS algorithm is that it could be easily extended to generate the data association hypotheses in the measurement-oriented approach [2].

Three algorithms which are suitable for implementation in a multiprocessor system are also developed. In the three algorithms, the computation of the *a posteriori* probabilities, β_j^t's is not based on the generation of the data association hypotheses like in the DFS algorithm. Each β_j^t in the three algorithms is decomposed into two parts. The computation of one part is trivial and the various ways of computing the other part (due to interference from other targets) lead to the development of the three algorithms.

In the first algorithm, the computation of the interference part of β_j^t is performed recursively in a top-to-bottom mode. In the worst case, this recursive algorithm requires more multiplications and additions than the fast JPDA algorithm [1]. However, in the average case, the recursive algorithm is expected to perform much better than the fast JPDA algorithm. Furthermore, the recursive algorithm requires less storage space than the fast JPDA algorithm. In comparison with the DFS algorithm developed earlier in this research, the recursive algorithm requires less multiplications and additions but more storage space than the DFS algorithm. The most important feature of the recursive algorithm is that it can be implemented on a multiprocessor system, which could speed up the computation of the β_j^t's significantly. On the other hand, the recursive algorithm is not as suitable as the DFS algorithm for extension to the measurement-oriented approach since it is not based on the generation of data association hypotheses.

The second algorithm is a nonrecursive algorithm. Since the interference part of β_j^t is computed by polynomial multiplication, the computational cost is drastically reduced with only a slight increase in memory when the density of targets is not very high. The nonrecursive algorithm is especially suitable for a small tracking system where the

computational capability is limited.

In the last algorithm, only the interference from the neighboring targets of target t is considered in the computation of the β_j^t's. The computational cost and memory requirement are only marginally increased in this approximating algorithm when it is compared with the PDAF [3]. It is apparent from the computer simulation that performance of the approximating algorithm is quite close to that of the original JPDAF in most of the scenarios presented in the simulation. Generally speaking, the approximating algorithm is proposed for applications where both the computational efficiency and the memory requirement are critical.

Another issue, which is quite different from data association as discussed above, in multitarget tracking is brought to attention. In the original JPDAF, the accuracy of the target state estimation is affected not only by the measurement noise but also by the uncertainty in the origins of the measurements. To improve the accuracy of the target state estimation, a JPDAS (JPDA smoothing) algorithm is proposed. Some computer simulation results have been presented to illustrate the improvement in tracking accuracy achieved by introducing a small time delay between the instants of estimation and the latest measurement. The experimental results show that the JPDAS algorithm works well for tracking multiple targets in a cluttered environment. However, the introduction of the time lag causes an increase in the computational and storage requirements. Fortunately, a small lag produces a significant improvement in the estimation accuracy so that the increase in the computational burden is kept to a minimum.

In addition to the development of the algorithms for data association and target state estimation, a comprehensive analysis of the neural network solution to the data association problem in [4] and [5] is also provided. The analysis reveals that the Hopfield neural networks developed in [4] and [5] have improper energy functions. This resulted from misinterpretations of the properties of the JPDAF which the networks were designed to emulate and the improper selections of the constant coefficients in the energy functions. The computer simulation system developed for multitarget tracking in a cluttered environment is available along with the simulation results from the system. This simulation system consists of six modules, spanning system specification, measurement generation, clustering, data association, target state estimation, and run-time monitoring.

References

[1] J. L. Fisher and D. P. Casasent, "Fast JPDA multitarget tracking algorithn," Applied Optics, vol. 28, January 1989, pp. 371-376.

[2] D. B. Reid, "An algorithm for tracking multiple targets," IEEE Trans. Automatic Control, vol. 24, December 1979, pp. 843-854.

[3] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," Automatica, vol. 11, Sept. 1975, pp. 451-460.

[4] D. Sengupta and R. A. Iltis, "A neural-network solution to the data association problem in multitarget target tracking," IEEE Trans. Aerospace and Electronic Systems, vol. 25, January 1989, pp. 96-108.

[5] D. Sengupta and R. A. Iltis, "Multiple maneuvering target tracking using neural networks," Tech. Rept. 90, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, December 1990.

3. The Most Significant Research Accomplishment

The most significant research accomplishment is the incorporation of image-based tracking with errors in measurement as well as errors in registration, which can be coupled with the efficient data association algorithm developed for the target-oriented approach (and which is suitable for generalization to the measurement-oriented, trak-oriented, and multiple correlation approaches) for fast and accurate tracking of nonmaneuvering as well as maneuvering multitargets in clutter. The image processing algorithms are necessary to identify targets in the presence of anticipated background noise (including earth, lunar, star backgrounds, complicated spacecraft structures, specular reflections) and blur (including atmospheric turbulence, motion, optical aberration). The areas of application include not only surveillance, but also autonomous rendezvous and capture, autonomous planetary landing systems, and satellite servicing. The sensors could be passive (video cameras, stereo vision (3-D sensor)), as well as active (laser radar) and the benefits of image-based tracking systems could include economy (cheaper tracking systems) in addition to speed and accuracy.

4. Research Personnel

(a) Dr. N. K. Bose, HRB-Systems Professor of Electrical Engineering and Director of The Spatial and Temporal Signal Processing Center at The Pennsylvania State University, University Park, PA 16802.

Was appointed Principal Investigator of the project, originally under the supervision of Professor A. K. Mahalanabis, who passed away on July 31, 1988. Dr. Bose supervised the original project, whose funding expired in December 1988, and prepared a new proposal entitled, "Multitarget Tracking in Clutter: New Approaches," which was approved for funding for a 3-year period, effective March 1989. A no-cost extension permitted extension of the contract (Number N00014-86-K-0542) to May 15, 1992. During the period of the contract, Dr. Bose supervised the research, provided new ideas for implementing data association efficiently, and prepared manuscripts for submissionto and subsequent publication in reviewed journals. Dr. Bose also interacted with Professor Leon H. Sibul, of the Applied Research Laboratory at The Pennsylvania State University, and other scientists working on multitarget tracking, especially in the SDIO/IST Program. He regularly reported the progress of research at the annual SDIO/IST Workshop at Arlington, Virginia, where other principal investigators also came to present their ideas and share research results.

(b) Bin Zhou, post-doctoral scholar at The Center for Multivariate Analysis and The Spatial and Temporal Signal Processing Center at The Pennsylvania State University, University Park, PA 16802.

Was continuously supported as a graduate research assistant until the completion of his Ph.D. dissertation in late 1991. This dissertation is on very high demand and requests for copies of the dissertation have been received from scientists from all over the world, including Indonesia, Italy, and Sweden. Bin Zhou has developed a complete computer simulation system to simulate multiple target tracking in a cluttered environment.

5. Publications

(From previous ONR Contract No. N0014-86-K-0542-P004, July 1, 1989 to May 15, 1992)

1. N. K. Bose, H. C. Kim, and H. M. Valenzuela, "Recursive implementation of total least squares algorithm for image reconstruction," IEEE Transactions on Image Processing, under review.

2. N. K. Bose and L. H. Sibul, "Multidimensional Signal Processing: Sensor Array Processing," invited chapter in Handbook of Electrical Engineering, ed. R. C. Dorf, CRC Press, scheduled to appear in 1993.

3. B. Zhou and N. K. Bose, "A Comprehensive Analysis of the 'Neural Solution to the Multitarget Tracking Data Association Problem'," IEEE Trans. on Aerospace and Electronic Systems, accepted for publication and scheduled to appear in late 1992.

4. B. Zhou and N. K. Bose, "Multitarget Tracking in Clutter: Fast Algorithms for Data Association," IEEE Trans. on Aerospace and Electronic Systems, accepted for publication and scheduled to appear in mid-1993.

5. A. K. Mahalanabis, B. Zhou, and N. K. Bose, "Improved Multi-Target Tracking in Clutter by PDA Smoothing," IEEE Trans. on Aerospace and Electronic Systems, Vol. 26, January 1990, pp. 113-121.

6. A. K. Mahalanabis and B. Zhou, "A Joint Probabilistic Data Association Smoothing Algorithm for Multitarget Tracking," the 1988 American Control Conference, Atlanta, GA, June 1988, pp. 360-365.

Ph.D. Dissertation

B. Zhou, "Multitarget Tracking in a Cluttered Environment," Ph.D. Dissertation, Department of Electrical and Computer Engineering, The Pennsylvania State University, University Park, PA, 1991.

6. Selected preprints

EDICS:1.10 (Interpolation and Spatial Transformations) of IEEE Transactions on Image Processing

Recursive Implementation of Total Least Squares Algorithm for Image Reconstruction From Noisy, Undersampled Multiframes *

N. K. Bose, H. C. Kim, and H. M. Valenzuela

Department of Electrical Engineering
The Spatial and Temporal Signal Processing Center
The Pennsylvania State University
University Park, PA 16802
Tel:(814) 865-3912
Fax:(814) 865-7065

ABSTRACT †

It is shown how the efficient recursive total least squares algorithm recently developed by C. E. Davila for real data can be applied to image reconstruction from noisy, undersampled multiframes when the displacement of each frame relative to a reference frame is not accurately known. To do this, the complex-valued image data in the wavenumber domain is transformed into an equivalent real data problem to which Davila's algorithm is successfully applied. Two detailed illustrative examples are provided in support of the procedure.

* This research was partially supported by SDIO/IST and managed by the Office of Naval Research under Contract N00014-86-K-0542.

† Permission to publish this abstract separately is granted.

1. Introduction

In many applications, multiple low resolution images of an object are available, but a high resolution image is desired. For example, a camera aboard a satellite is used to take several pictures of an area on the ground and to transmit a sequence of undersampled low resolution frames, which are shifted from each other. Futhermore, those frames are often degraded versions of the original scene due to blur and noise. Through the task of image registration, the displacement of each frame relative to an arbitrarily chosen reference frame is measured. The type of blur and the characteristic of noise can be determined from the understanding of the various physical processes involved in the image formation. Subsequently, interpolation, deblurring, and filtering are required to reconstruct a high resolution image.

S. P. Kim, N. K. Bose, and H. M. Valenzuela recently developed an efficient algorithm to reconstruct a high resolution noise-free image from undersampled low resolution noisy multiframes [1]. This algorithm implements, simultaneously, the tasks of interpolation and noise filtering of the input images by applying the recursive least squares (RLS) sequential estimation theory in the wavenumber domain.

In [1], the displacements of the images are assumed to be known. Actually, errors occur in the estimation of the relative displacement during image registration. To combat this problem, we apply the Total Least Squares (TLS) method. This method is known to be very useful for improving the solution accuracy when errors are present not only in the observation but also in the measurement matrix [2]. A Recursive Total Least Squares (RTLS) algorithm was developed by C. E. Davila [3] in the context of the adaptive filtering problem.

In Section 2, the image interpolation model which is formulated by using the aliasing property of the Discrete Fourier Transform (DFT) is briefly described. The algorithm for high resolution image reconstruction, described in Section 3, is obtained by transforming the original complex-data problem to an equivalent real data problem to which the RTLS algorithm in [3] is applied. In Section 4, the performance of the algorithm is verified by computer simulations.

2. Reconstruction Model of High Resolution Images

The k-th $(M \times N)$ undersampled observed image frame, $f_k(i,j)$, $i = 0,1,\ldots,$ $M-1$, $j = 0,1,\ldots,N-1$, of the noise-free original continuous image, $f(x,y)$, can be written as

$$f_k(i,j) = f(iT_x + \delta_{xk}, jT_y + \delta_{yk}), \quad k = 1,2,...,p \tag{1}$$

where δ_{xk} and δ_{yk} are the estimated shifts and T_x and T_y are the sampling periods along the x and y axes, respectively [1]. We assume that the Fourier transform, $F^c(u,v)$ of the original continuous image $f(x,y)$ is approximated by the bandlimited constraint

$$|F^c(u,v)| = 0, \ |u| > L_x w_x \text{ and } |v| > L_y w_y \tag{2}$$

for some finite integers L_x and L_y where $w_x = (2\pi/T_x)$ and $w_y = (2\pi/T_y)$.

The multiframe image restoration model given in [1] can be written in the form

$$Z_k = Y_k^t \underline{F} + N_k \tag{3}$$

where the superscript t denotes the transpose operation, Z_k is the value at the wavenumber point (m,n) of the DFT of the k-th noisy undersampled frame, N_k is

the corresponding noise term in the wavenumber domain, $\underline{F}$ is a vector containing the $p = 4L_xL_y$ interpolated components at wavenumber point (m,n) denoted by

$$\begin{aligned}\underline{F} &= \left[F_{mn}(1),\ F_{mn}(2),\ldots,F_{mn}(p)\right] \\ &\triangleq [\hat{f}_1,\ \hat{f}_2,\ldots,\hat{f}_p]\end{aligned} \tag{4}$$

and the coefficient vector Y_k is defined as

$$Y_k^t \triangleq [\phi_{k,1},\ \phi_{k,2},\ \ldots,\phi_{k,p}] \tag{5}$$

where

$$\phi_{k,r} = \frac{1}{T_xT_y} exp\Big\{j2\pi\big\{\delta_{xk}(\frac{m}{MT_x} + \frac{l_x}{T_x}) + \delta_{yk}(\frac{n}{NT_y} + \frac{l_y}{T_y})\big\}\Big\} \tag{6}$$

after defining the lexicographical ordering associated with row-wise scanning by

$$l_x \triangleq (r-1)mod(2L_x) - L_x \quad \text{and} \quad l_y \triangleq \lfloor (r-1)/(2L_x) \rfloor - L_y. \tag{7}$$

If there are J frames available($J \geq 4L_xL_y$), then eqn. (3) may be used to obtain the matrix equation

$$\underline{Z}_J = \underline{\Phi}_J\underline{F} + \underline{N}_J \tag{8}$$

where

$$\underline{\Phi}_J = [Y_1, Y_2, \ldots, Y_J]^t$$

$$\underline{Z}_J = [Z_1, Z_2, \ldots, Z_J]^t$$

$$\underline{N}_J = [N_1, N_2, \ldots, N_J]^t.$$

By solving eqn. (8), we obtain the high resolution samples at the generic wavenumber point (m,n). The interpolating factors are, respectively, $2L_x$ and $2L_y$

along the x and y axes. The inverse DFT of the interpolated image gives the high resolution image.

3. Reconstruction of High Resolution Image by RTLS Algorithm

The TLS algorithm provides a solution when there are errors in both the observation vector and in the measurement matrix. Prior to [3], the solution was not generated recursively. In [3], an algorithm was developed for the linear regression problem to produce unbiased filter coefficients for FIR adaptive filters. To be able to apply the result in [3] to the image reconstruction problem under consideration here (with both observation noise and inter-frame displacement error present) it is necessary to follow the procedure described below.

To account for the errors in estimation of the shifts during the registration phase, we define,

$$\begin{aligned} \hat{\delta}_{xk} &\triangleq \delta_{xk} + \triangle\delta_{xk} \\ \hat{\delta}_{yk} &\triangleq \delta_{yk} + \triangle\delta_{yk}, \end{aligned} \tag{9}$$

where δ_{xk} and δ_{yk} are the actual shifts and $\triangle\delta_{xk}$, $\triangle\delta_{yk}$ are the respective displacement errors along the x and y axes in the registration process. Substituting eqn. (9) into eqn. (6), we obtain

$$\hat{\phi}_{kr} = \frac{1}{T_x T_y} exp\Big\{ j2\pi\big\{(\delta_{xk}+\triangle\delta_{xk})(\frac{m}{MT_x}+\frac{l_x}{T_x})+(\delta_{yk}+\triangle\delta_{yk})(\frac{n}{NT_y}+\frac{l_y}{T_y})\big\}\Big\}. \tag{10}$$

For small magnitudes of $\triangle\delta_{xk}$ and $\triangle\delta_{yk}$, $\hat{\phi}_{kr}$ in eqn. (10) may be approximated by

$$\hat{\phi}_{kr} \simeq \phi_{kr} + \epsilon_{kr} \tag{11}$$

where ϕ_{kr}, defined in eqn. (6), is expressible as

$$\phi_{kr} = \hat{\phi}_{kr}\Big|_{\triangle\delta_{xk}=\triangle\delta_{yk}=0}$$

and

$$\epsilon_{kr} = j2\pi\{\triangle\delta_{xk}(\frac{m}{MT_x} + \frac{l_x}{T_x}) + \triangle\delta_{yk}(\frac{n}{NT_y} + \frac{l_y}{T_y})\}\phi_{kr} \tag{12}$$

for

$$\left|2\pi\{\triangle\delta_{xk}(\frac{m}{MT_x} + \frac{l_x}{T_x}) + \triangle\delta_{yk}(\frac{n}{NT_y} + \frac{l_y}{T_y})\}\right| \ll 1. \tag{13}$$

Since the maximum values of m and n are, respectively, $M-1$ and $N-1$, the inequality in eqn. (13) has only to be checked for the case when $m = M-1$ and $n = N-1$. Substituting eqn. (7) in eqn. (13), simple sufficient conditions for the bound in eqn. (13) to hold are

$$|\triangle\delta_{xk}| \ll \frac{T_x}{4\pi(L_x+1)} = \frac{w_x}{2(L_x+1)} \tag{14a}$$

and

$$|\triangle\delta_{yk}| \ll \frac{T_y}{4\pi(L_y+1)} = \frac{w_y}{2(L_y+1)}. \tag{14b}$$

Using eqns. (8) and (11) and after replacing the subscript J by k, we get

$$\underline{Z}_k = [\underline{\Phi}_k + \underline{E}_k]\underline{F} + \underline{N}_k \tag{15}$$

where the $(i,j)th$ element of the matrix $\underline{E}_k$ is ϵ_{ij} of eqn. (12). The matrix $\underline{E}_k$ in eqn. (15) is basically a perturbation of the coefficient matrix $\underline{\Phi}_k$ produced by the delay estimation errors. We now consider the TLS problem formulated as

$$\begin{aligned} &\text{minimize } \|[\underline{N}_k \vdots \underline{E}_k]\| \\ &\text{subject to } \underline{Z}_k - \underline{N}_k = [\underline{\Phi}_k + \underline{E}_k]\underline{F} \end{aligned} \tag{16}$$

where $||\cdot||$ denotes the Frobenius norm

$$||B||^2 = \sum_i \sum_j |b_{ij}|^2$$

on the matrix $B = [b_{ij}]$. Define

$$\hat{\underline{\Phi}}_k \stackrel{\triangle}{=} [\underline{Z}_k \vdots \underline{\Phi}_k] \tag{17}$$

$$\underline{W}_k \stackrel{\triangle}{=} [-\underline{N}_k \vdots \underline{E}_k] \tag{18}$$

and

$$q_k \stackrel{\triangle}{=} \begin{pmatrix} 1 \\ -\underline{F} \end{pmatrix} = [1, -\hat{f}_1, -\hat{f}_2, \ldots, -\hat{f}_p]^t. \tag{19}$$

The TLS problem in eqn. (16) can be restated as:

$$\begin{gathered} \text{minimize } ||\underline{W}_k|| \\ \text{subject to } [\hat{\underline{\Phi}}_k + \underline{W}_k]q_k = \underline{0}. \end{gathered} \tag{20}$$

From what is known in [4], the above constrained minimization problem can be associated with the equivalent minimization problem

$$\mu(q_k) = \min ||\underline{W}_k||^2 = \min_{q_k} \frac{q_k^H \hat{\underline{\Phi}}_k^H \hat{\underline{\Phi}}_k q_k}{q_k^H q_k} = \min_{q_k} \frac{q_k^H R_k q_k}{q_k^H q_k}, \tag{21}$$

where the superscript H denotes the complex conjugate transpose operation, referred to as Hermitian conjugation. The Hermitian matrix R_k in eqn. (21) is defined as

$$R_k \stackrel{\triangle}{=} \hat{\underline{\Phi}}_k^H \hat{\underline{\Phi}}_k = \sum_{j=1}^{k} r_j^* r_j^t \tag{22}$$

where the star superscript denotes the complex conjugate operation and

$$r_k \stackrel{\triangle}{=} [Z_k,\ \phi_{k1} + \varepsilon_{k1},\ \phi_{k2} + \varepsilon_{k2}, \ldots, \phi_{kp} + \varepsilon_{kp}]^t. \tag{23}$$

The minimization problem in eqn. (21) corresponds to the obtaining of the eigenvector q_k associated with the smallest eigenvalue of R_k. Given the previous eigenvector q_{k-1}, we update it to obtain q_k from

$$q_k = q_{k-1} + \alpha_k \psi_k, \tag{24}$$

as in [5], where ψ_k is a correction vector, chosen in [3] to be the Kalman gain vector,

$$\psi_k = K_k = R_k^{-1} r_k \tag{25}$$

and the scalar α_k is derived from the minimization of

$$\eta(q_k) = \eta(q_{k-1} + \alpha_k \psi_k) = \frac{q_k^H R_k q_k}{q_k^H q_k}. \tag{26}$$

In order to update the Kalman gain, in our formulation we use the matrix inversion lemma in the RLS algorithm. To get q_k recursively, as in eqns. (24)-(26), we provide a mechanism for transforming our problem involving complex variables to an equivalent problem involving real variables where C. E. Davila's algorithm [3] directly applies. To do this, let $\Re(x)$ and $\Im(x)$ denote, respectively, the real and imaginary parts of x. Then express R_k and q_k as

$$R_k = A_k + jB_k \tag{27}$$

where

$$A_k = \Re(R_k) \quad \text{and} \quad B_k = \Im(R_k),$$

and

$$q_k = c_k + jd_k \tag{28}$$

where

$$c_k = \Re(q_k) \text{ and } d_k = \Im(q_k).$$

Since R_k is Hermitian, therefore $A_k = A_k^t$ and $B_k^t = -B_k$. Define

$$G_k \triangleq \begin{pmatrix} A_k & B_k \\ B_k^t & A_k \end{pmatrix} \tag{29}$$

and

$$h_k \triangleq \begin{pmatrix} c_k \\ d_k \end{pmatrix}. \tag{30}$$

Then it can be verified that $\eta(q_k)$ in eqn. (26) is expressible as

$$\eta(q_k) = \frac{q_k^H R_k q_k}{q_k^H q_k} = \frac{h_k^t G_k h_k}{h_k^t h_k}. \tag{31}$$

If we express the "Kalman gain" K_k in our problem as

$$K_k = V_k + jW_k \tag{32}$$

where

$$V_k = \Re(K_k) \text{ and } W_k = \Im(K_k),$$

then an associated vector M_k is defined below.

$$M_k \triangleq \begin{pmatrix} V_k \\ W_k \end{pmatrix}. \tag{33}$$

Given h_{k-1}, the updated vector h_k in eqn. (30) is

$$h_k = h_{k-1} + \beta_k M_k, \tag{34}$$

where the scalar β_k is obtained by minimizing

$$\eta(h_k) = \eta(h_{k-1} + \beta_k M_k) = \frac{h_k^t G_k h_k}{h_k^t h_k}. \tag{35}$$

To update G_k in eqn. (29), we need to update A_k and B_k from r_k. We define

$$s_k = \Re(r_k) \quad \text{and} \quad e_k = \Im(r_k). \tag{36}$$

It can be easily shown that

$$A_k = A_{k-1} + s_k s_k^t + e_k e_k^t \tag{37}$$

$$B_k = B_{k-1} + s_k e_k^t - e_k s_k^t. \tag{38}$$

Substituting eqn. (34) into eqn. (35) and differentiating with respect to β, we obtain the quadratic equation

$$a\beta^2 + b\beta + c = 0 \tag{39}$$

where

$$\begin{aligned} a &= h_{k-1}^t G_k \phi_k \phi_k^t \phi_k - \phi_k^t G_k \phi_k h_{k-1}^t \phi_k \\ b &= h_{k-1}^t G_k h_{k-1} \phi_k^t \phi_k - \phi_k^t G_k \phi_k \\ c &= h_{k-1}^t G_k h_{k-1} h_{k-1}^t \phi_{k-1} - h_{k-1}^t G_k \phi_k, \end{aligned} \tag{40}$$

as in [3]. It has been shown in [5] that eqn. (39) has only real roots. Choosing the smallest root of this equation, we obtain the updated vector h_k from eqn. (34). We now summarize the algorithm below.

MODIFIED ALGORITHM FOR COMPLEX DATA

$$\phi_k = M_k$$

$$A_k = A_{k-1} + s_k s_k^t + e_k e_k^t$$

$$B_k = B_{k-1} + s_k e_k^t - e_k s_k^t$$

$$G_k = \begin{pmatrix} A_k & B_k \\ B_k^t & A_k \end{pmatrix}$$

$$a = h_{k-1}^t G_k \phi_k \phi_k^t \phi_k - \phi_k^t G_k \phi_k h_{k-1}^t \phi_k$$

$$b = h_{k-1}^t G_k h_{k-1} \phi_k^t \phi_k - \phi_k^t G_k \phi_k$$

$$c = h_{k-1}^t G_k h_{k-1} h_{k-1}^t \phi_{k-1} - h_{k-1}^t G_k \phi_k$$

$$\beta_k = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$$h_k = h_{k-1} + \beta_k \phi_k$$

$$h_k = \frac{h_k}{\sqrt{h_k^t h_k}}$$

The final solution can be obtained by converting h_k to q_k and

$$\hat{f}_{i-1} = -\frac{(q_k)_i}{(q_k)_1}; \quad i = 2, \ldots, p \tag{41}$$

where $(q_k)_i$ is the i-th element of the vector q_k.

4. Computer Simulation

In this computer simulated examples, a high resolution (256×256) image is recursively reconstructed from a set of low resolution (128×128) noisy frames which are shifted with respect to a reference frame. These shifts or displacements are not accurately known. In order to reduce the size of the arrays to be processed, the

input image was partitioned into 16 nonoverlapping sections each of size (32×32) and the recursive Total Least Squares (RTLS) algorithm described in Section 3 was independently applied to each such section. It is recalled that in [1] a similar approach was adopted but instead of RTLS the recursive least squares (RLS) algorithm was used. For each one of these 16 sections, the interpolation problem corresponds to the reconstruction of a (64×64) image from a sequence of shifted low resolution (32×32) noisy input frames when the inter-frame displacements are not accurately known. To generate the $K = 16$ shifted low resolution input frames required in the simulation from the available data, we use the DFT-based interpolation technique described in [1]. After assigning one of the input frames to be the reference frame, we label it as frame number 1 and the remaining ones are sequentially labelled from $k = 2$ to $k = 16$. The relative shifts of these frames with respect to the reference one along the x and y axes are denoted, respectively, by δ_{xk} and δ_{yk} for $k = 2, 3, \ldots, 16$.

To simulate the errors $\triangle\delta_{xk}$ and $\triangle\delta_{yk}$, $k = 2, 3, \ldots, 16$, in the shifts in eqn. (9), we use the expression in eqn. (12) after assigning to these displacement errors uniformly distributed random values over the interval $[-\frac{1}{8}, \frac{1}{8}]$. Subsequently, an uniformly distributed zero mean noise was added to each input frame. These noise sequences were statistically independent and the signal-to-noise ratio (SNR) was chosen to be 20dB. The first simulation example uses the image of a girl given in Fig. 1. Fig. 2 shows 4 (out of a total of 16) low resolution (128×128) noisy input frames, each having a displacement error with respect to the chosen reference frame (naturally, the reference frame is defined to have zero displacement error), and an additive noise with SNR level of 20dB. The RTLS algorithm given in Section 3 was

used to recursively obtain new estimates of the reconstructed high resolution image. The last estimate (the sixteenth) of this sequence is shown in Fig. 3. The visual quality of the reconstructed image in Fig. 3 is very close to the original image in Fig. 1.

To illustrate the intermediate steps of the reconstruction, we provide in Fig. 4 a generic set of 16 (32×32) frames corresponding to a section of the low resolution noisy image. From this set, a (64×64)-section image is obtained by application of the RTLS algorithm. The sequence of estimates leading to the construction of the high resolution filtered image is shown in Fig. 5. As expected, the first few iterations of the RTLS algorithm provide poor estimates because the algorithm is in a transient stage. Subsequently, the estimates improve and the algorithm could be terminated even before the sixteenth iteration without seriously affecting the visual quality of the reconstructed image.

The second simulation example uses the image of an aerial photograph given in Fig. 6. Fig. 7 shows 4 (out of a total of 16) low resolution (128×128) noisy input frames, each having a displacement error with respect to the chosen reference frame, and an additive noise with SNR level of 20dB. The last estimate (the sixteenth) obtained by application of the RTLS algorithm is shown in Fig. 8. Fig. 9 shows a generic set of 16 (32×32) frames corresponding to a section of the low resolution noisy image. From this set, a (64×64)-section image is obtained by application of the RTLS algorithm. The sequence of estimates leading to the construction of the high resolution filtered image for this section is shown in Fig. 10.

5. Conclusion

It is shown how the total least squares recursive algorithm for the real data FIR adaptive filtering problem may be applied to reconstruct a high resolution filtered image from undersampled, noisy, multiframes, when the inter-frame displacements are not accurately known. This is done in the wavenumber domain after transforming the complex data problem to an equivalent real data problem, to which the algorithm developed in [3] applies. The procedure developed also applies to the case when the multiframes are degraded by linear shift-invariant blurs, in a manner similar to that achieved in [1], where the least squares sequential estimation theory was applied. All the advantages regarding implementation via massively parallel computational architecture apply here as in [1], because of the decoupling resulting from the type of processing carried out in the wavenumber domain – a feature also in evidence in a different but related context [6].

References

[1] S. P. Kim, N. K. Bose, and H. M. Valenzuela, "Recursive Reconstruction of High Resolution Image From Noisy Undersampled Frame," *IEEE Transaction on Acoustics, Speech, and Signal Processing*, Vol. 38, No. 6, pp. 1013-1027, June 1990.

[2] S. Van Huffel and J. Vandewalle, The Total Least Squares Problem : Computational Aspects and Analysis, *SIAM*, Philadelphia, 1991.

[3] C. E. Davila, "Recursive Total Least Squares Algorithms for Adaptive Filtering," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toronto, Canada, pp. 1853-1856, 1991. Additional details in C. E. Davila, "An Efficient Recursive Total Least Squares Algorithms for FIR Adaptive Filtering," submitted to *IEEE Transaction on Signal Processing,* and provided by the author to N. K. Bose.

[4] G. H. Golub, "Some Modified Matrix Eigenvalue Problems," *SIAM Review*, Vol. 15, No.2, pp. 318-334, April 1973.

[5] I. Shavitt, C. F. Bender, A. Pipano, and R. P. Hosteny, "The Iterative Calculation of Several of the Lowest or Highest Eigenvalues and Corresponding Eigenvectors of Very Large Symmetric Matrices," *Journal of Computational Physics* , Vol. 11, pp. 90-108, 1973.

[6] C. C. Kuo and B. C. Levy, "Discretization and Solution of Elliptic PDE's – a digital signal processing approach," *Proceeding of IEEE*, Vol. 78, pp. 1808-1842, December 1990.

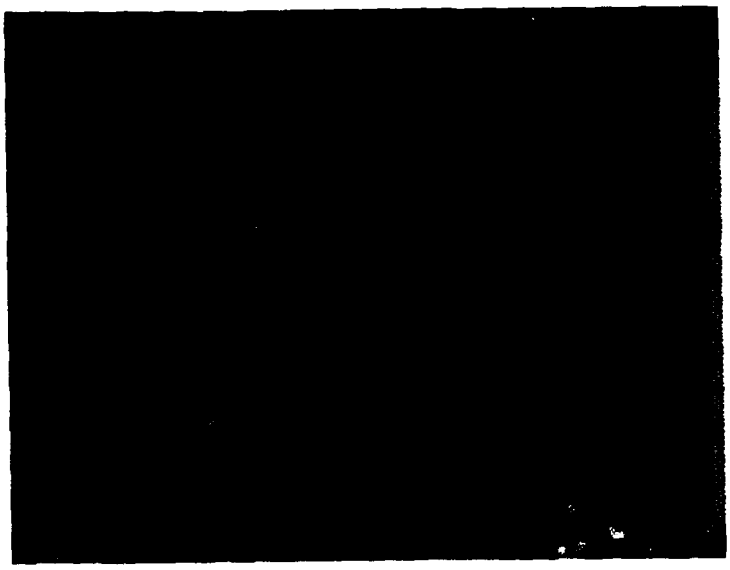


Fig. 1. Original image - a girl

Fig. 2. 1st - 4th (128 × 128) input frames
($\pm\frac{1}{8}$ pixel displacement error and 20dB SNR)

Fig. 3. The final (256×256) reconstructed image by RTLS (after 16 frames are processed.)

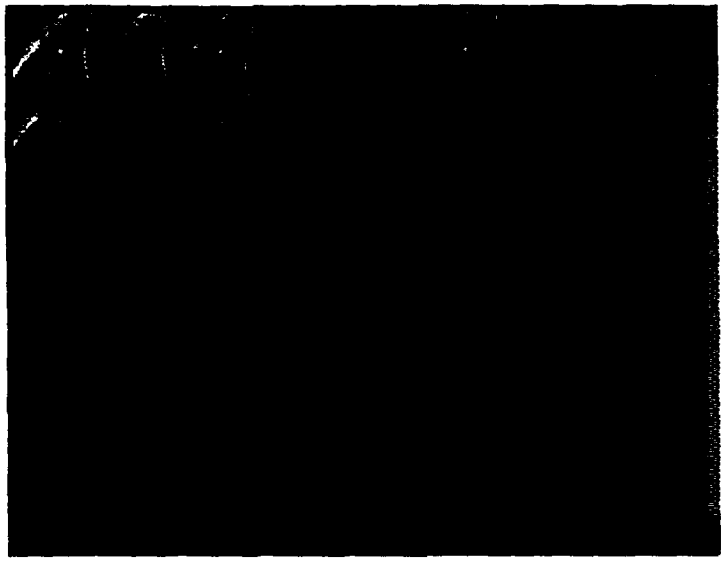


Fig. 4. 16 (32×32) input frames
($\pm\frac{1}{8}$ pixel displacement error and 20dB SNR)

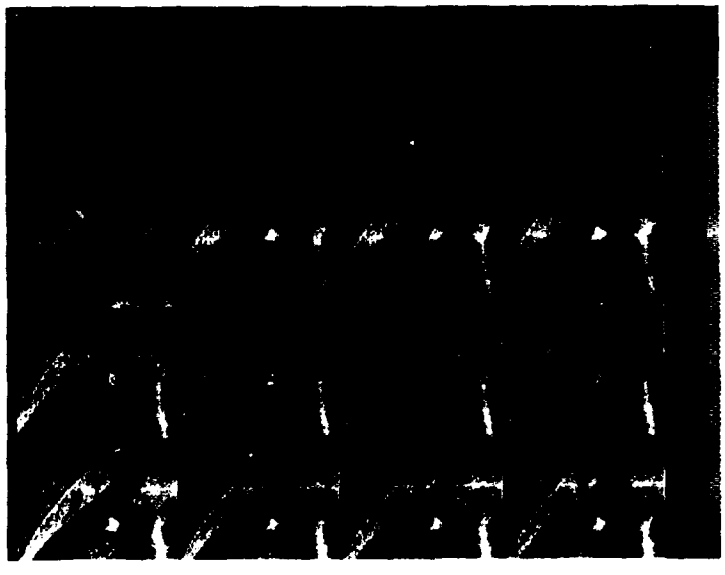


Fig. 5. ($64 \cdot 64$) reconstructed image by RTLS
(each output is obtained recursively.)

Fig. 6. Original image - Aerial photograph

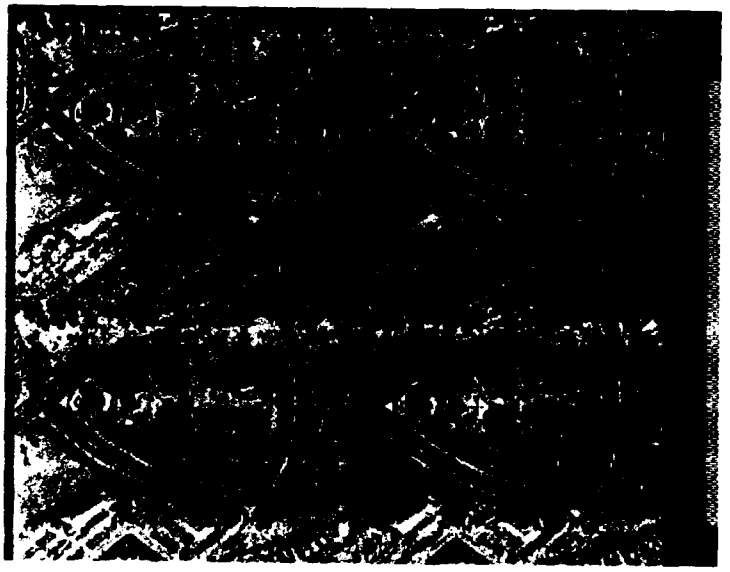


Fig. 7. 1st - 4th (128 × 128) input frames
($\pm\frac{1}{8}$ pixel displacement error and 20dB SNR)

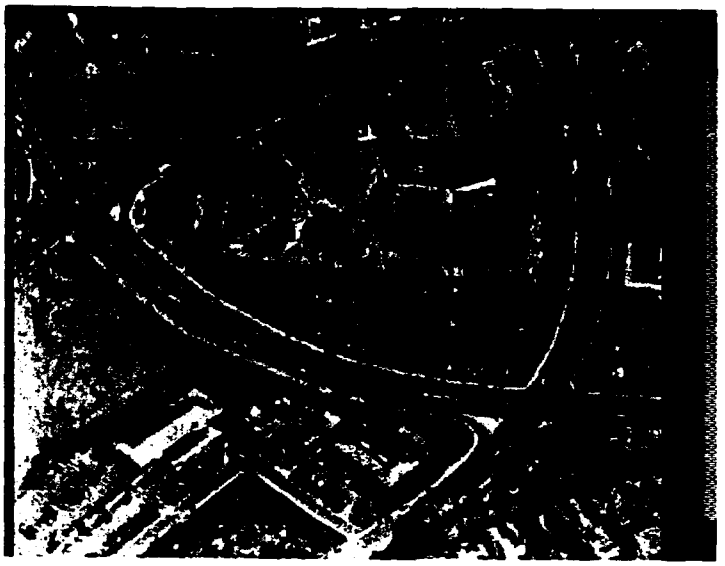


Fig. 8. The final (256 × 256) reconstructed image by RTLS (after 16 frames are processed.)

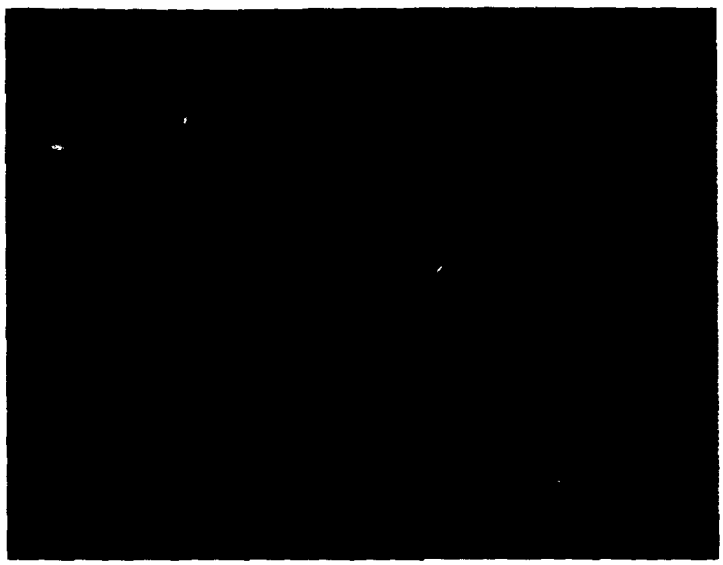


Fig. 9. 16 (32×32) input frames
($\pm\frac{1}{8}$ pixel displacement error and 20dB SNR)

Fig. 10. (64×64) reconstructed image by RTLS
(each output is obtained recursively.)

To appear in The Electrical Engineering Handbook, January 1993.

MULTIDIMENSIONAL SIGNAL PROCESSING: SENSOR ARRAY PROCESSING

N. K. Bose and L. H. Sibul, The Pennsylvania State University

Introduction

Multidimensional signal processing tools apply to aperture and sensor array processing. Planar sensor arrays can be considered to be sampled apertures. Three dimensional or volumetric arrays can be viewed as multidimensional spatial filters. Therefore, the topics of sensor array processing, aperture processing, and multidimensional signal processing can be studied under a unified format. The basic function of the receiving array is transduction of propagating waves in the medium into electrical signals. Propagating waves are fundamental in radar, communication, optics, sonar, and geophysics. In electromagnetic applications, basic transducers are antennas and arrays of antennas. A large body of literature that exists on antennas and antenna arrays can be exploited in the areas of aperture and sensor array processing. Much of the antenna literature deals with transmitting antennas and their radiation patterns. Due to the reciprocity of transmitting and receiving transducers, key results that have been developed for transmitters can be used for analysis of receiver aperture and/or array processing. Transmitting transducers radiate energy in desired directions, whereas receiving apertures/arrays act as spatial filters that emphasize signals from a desired look direction while discriminating against interferences from other directions. The spatial filter wavenumber response is called the receiver beampattern. Transmitting apertures are characterized by their radiation patterns.

Conventional beamforming deals with the design of fixed beampatterns for given specifications. Optimum beamforming is the design of beampatterns to meet a specified optimization criterion. It can be compared to optimum filtering, detection, and estimation. Adaptive beamformers sense their operating environment (for example, noise covariance matrix) and adjust beamformer parameters so that their performance is optimized [Monzingo and Miller, 1980]. Adaptive beamformers can be compared with adaptive filters.

Multidimensional signal processing techniques have found wide application in seismology, where a group of identical seismometers, called seismic arrays, are used for event location, studies of the earth's sedimentation structure, and separation of coherent signals from noise, which, sometimes, may also propagate coherently across the array but with different horizontal velocities, by employing velocity filtering [Claerbout, 1976]. Velocity filtering is performed by multidimensional filters, and allows also for the enhancement of signals which may occupy the same wavenumber range as noise or undesired signals do. In a broader context, beamforming can be used to separate signals received by sensor arrays based on frequency, wavenumber, and velocity (speed as well as direction) of propagation. Both the transfer and unit impulse-response functions of a velocity filter are two-dimensional functions in the case of one-dimensional arrays. The transfer function involves frequency and wavenumber (due to spatial sampling by equally spaced sensors) as independent variables, whereas the unit impulse response depends upon time and location within the array. Two-dimensional filtering is not limited to velocity filtering by means of seismic array. Two-dimensional spatial filters are frequently used, for example, in the interpretation of gravity and magnetic maps to differentiate between regional and local features. Input data for these filters may be observations in the survey of an area conducted over a planar grid over the earth's surface. Two-dimensional wavenumber digital filtering principles are useful for this purpose. Velocity filtering by means of two-dimensional arrays may be accomplished by properly shaping a three-dimensional response function $H(k_1, k_2, \omega)$. Velocity filtering by three-dimensional arrays may be accomplished through a four-dimensional function $H(k_1, k_2, k_3, \omega)$ as explained below.

Spatial Arrays, Beamformers, and FIR Filters:

A propagating plane wave, $s(\underline{x},t)$, is, in general, a function of the three-dimensional space variables $(x_1,x_2,x_3) \triangleq \underline{x}$ and the time variable t. The 4-D Fourier transform of the stationary signal $s(\underline{x},t)$ is

$$S(\underline{k},\omega) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(\underline{x},t)\, e^{-j\left(\omega t - \sum_{i=1}^{3} k_i x_i\right)} dx_1 dx_2 dx_3 dt \tag{1}$$

which is referred to as the wavenumber-frequency spectrum of $s(\underline{x},t)$, and $(k_1,k_2,k_3) \triangleq \underline{k}$ denotes the wavenumber variables in radians per unit distance and ω is the frequency variable in radians per second. If c denotes the velocity of propagation of the plane wave, the following constraint must be satisfied

$$k_1^2 + k_2^2 + k_3^2 = \frac{\omega^2}{c^2}.$$

If the 4-D Fourier transform of the unit impulse response $h(\underline{x},t)$ of a 4-D linear shift-invariant (LSI) filter is denoted by $H(\underline{k},\omega)$, then the response $y(\underline{x},t)$ of the filter to $s(\underline{x},t)$ is the 4-D linear convolution of $h(\underline{x},t)$ and $s(\underline{x},t)$, which is, uniquely, characterized by its 4-D Fourier transform

$$Y(\underline{k},\omega) = H(\underline{k},\omega)\, S(\underline{k},\omega). \tag{2}$$

The inverse 4-D Fourier transform, which forms a 4-D Fourier transform pair with (1) is

$$s(\underline{x},t) = \frac{1}{(2\pi)^4}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(\underline{k},\omega)\, e^{j\left(\omega t - \sum_{i=1}^{3} k_i x_i\right)} dk_1 dk_2 dk_3 d\omega. \tag{3}$$

It is noted that $S(\underline{k},\omega)$ in (1) is product separable, i.e. expressible in the form

$$S(\underline{k},\omega) = S_1(k_1)\, S_2(k_2)\, S_3(k_3)\, S_4(\omega), \tag{4}$$

where each function on the right-hand side is a univariate function of the respective independent variable, if and only if $s(\underline{x},t)$ in (1) is also product separable. In beamforming, $S_1(k_1)$ in (4) would be the far-field beampattern of a linear array along the x_1-axis. For example, the normalized beampattern of a uniformly weighted (shaded) linear array of length L is

$$S(k,\theta) = \frac{\sin\left(\frac{kL\sin\theta}{2}\right)}{\left(\frac{kL}{2}\sin\theta\right)} \tag{5}$$

where $\lambda = \left(\frac{2\pi}{k}\right)$ is the wavelength of the propagating plane wave and θ is the angle of arrival at array site as shown in Figure 1. Note that θ is explicitly admitted as a variable in $S(k,\theta)$ to allow for the possibility that for a fixed wavenumber, the beampattern could be plotted as a function of the angle of arrival. In that case, when θ is zero, the wave impinges the array broadside and the normalized beampattern evaluates to unity.

The counterpart, in aperture and sensor array processing, of the use of

window functions in spectral analysis for reduction of sidelobes is the use of aperture shading. In aperture shading, one simply multiplies a uniformly weighted aperture by the shading function. The resulting beampattern is, then, simply the convolution of the beampattern of the uniformly shaded volumetric array and the beampattern of the shading function. Fourier transform relationship between the stationary signal $s(\underline{x},t)$ and the wavenumber frequency spectrum $S(\underline{k},\omega)$ allows one to exploit high resolution spectral analysis techniques for the high resolution estimation of the direction of arrival [Pillai, 1989].

Discrete Arrays for Beamforming:

An array of sensors could be distributed at distinct points in space in various ways. Line arrays, planar arrays, and volumetric arrays could be either uniformly spaced or nonuniformly spaced, including the possibility of placing sensors randomly according to some probability distribution function. Uniform spacing along each of the coordinate axis permits one to exploit the well-developed multidimensional signal processing techniques concerned with filter design, DFT computation via FFT and high resolution spectral analysis of sampled signals [Dudgeon, 1977]. Nonuniform spacing, sometimes, might be useful for reducing the number of sensors, which, otherwise, might be constrained to satisfy a maximum spacing between uniformly placed sensors to avoid grating lobes due to aliasing, as explained later. A discrete array, uniformly spaced, is convenient for the synthesis of a digital filter or beamformer by the performing of digital signal processing operations (namely delay, sum, and multiplication or weighting) on the signal received by a collection of sensors distributed in space. The sequence of the nature of operations dictate the types of beamformer. Common beamforming systems are of the straight summation, delay-and-sum, and weighted delay-and-sum types. The geometrical distribution of sensors and the weights w_i associated with each sensor are crucial factors in the shaping of the filter characteristics. In the case of a linear array of N equispaced sensors, which are spaced D units apart, starting at the origin $x_1=0$, the function

$$W(k_1) = \frac{1}{N}\sum_{n=0}^{N-1} w_n e^{-jk_1 nD} \tag{6}$$

becomes the array pattern, which may be viewed as the frequency response function for a finite impulse response (FIR) filter, characterized by the unit impulse response sequence $\{w_n\}$. In the case when $w_n=1$, (6) simplifies to

$$W(k_1) = \frac{1}{N}\frac{\sin\left(\frac{k_1 ND}{2}\right)}{\sin\left(\frac{k_1 D}{2}\right)}\exp\left\{-j\frac{(N-1)k_1 D}{2}\right\}. \tag{7}$$

If the N sensors are symmetrically placed on both sides of the origin, including one at the origin, and the sensor weights are $w_n=1$, then the linear array pattern becomes

$$W(k_1) = \frac{1}{N}\frac{\sin\frac{k_1 ND}{2}}{\sin\frac{k_1 D}{2}}.$$

For planar arrays, direct generalizations of the preceding linear array results can be obtained. To wit, if the sensors with unity weights are located at coordinates $(kD,\ell D)$, where $k=0,\pm1,\pm2,\ldots,\pm\left(\frac{N-1}{2}\right)$, and

$\ell = 0, \pm 1, \pm 2, \ldots, \pm\left(\frac{M-1}{2}\right)$, for odd integer values of N and M then the array pattern function becomes

$$W(k_1, k_2) = \frac{1}{NM} \sum_{k=-\left(\frac{N-1}{2}\right)}^{\left(\frac{N-1}{2}\right)} \sum_{\ell=-\left(\frac{M-1}{2}\right)}^{\left(\frac{M-1}{2}\right)} \exp\{-j(k_1 kD + k_2 \ell D)\}$$
$$= \frac{1}{NM} \frac{\sin\left(\frac{k_1 ND}{2}\right)}{\sin\left(\frac{k_1 D}{2}\right)} \frac{\sin\left(\frac{k_2 MD}{2}\right)}{\sin\left(\frac{k_2 D}{2}\right)} \tag{8}$$

Routine generalizations to 3-D spatial arrays are also possible. The array pattern functions for other geometrical distributions may also be routinely generated. For example, if unit weight sensors are located at the six vertices and the center of a regular hexagon each of whose sides is D units long, then the array pattern function can be shown to be

$$W(k_1, k_2) = \frac{1}{7}\left[1 + 2\cos k_1 D + 4\cos\frac{k_1 D}{2}\cos\frac{\sqrt{3}k_2 D}{2}\right]. \tag{9}$$

The array pattern function reveals how selective a particular beamforming system is. In the case of a typical array function shown in (7), the beamwidth, which is the width of the main lobe of the array pattern, is inversely proportional to the array aperture. Because of the periodicity of the array pattern function, the main lobe is repeated at intervals of $\frac{2\pi}{D}$. These repetitive lobes are called grating lobes, whose existence may be interpreted in terms of spatial frequency aliasing resulting from a sampling interval D due to the N receiving sensors located at discrete points in space. If the spacing D between sensors satisfies

$$D \leq \frac{\lambda}{2}, \tag{10}$$

where λ is the smallest wavelength component in the signal received by the array of sensors, then the grating lobes have no effect on the received signal. A plane wave of unit amplitude which is incident upon the array at bearing angle θ degrees as shown in Figure 1 produces outputs at the sensors given by the vector

$$\underline{s}_\theta = [\exp(j0) \;\; \exp(jk_1 D\sin\theta) \ldots \exp(jk_1(N-1)D\sin\theta)]^t \tag{11}$$

where $k_1 = \frac{2\pi}{\lambda}$ is the wavenumber. In array processing, the array output y_θ may be viewed as the inner product of an array weight vector $\underline{w}$ and the steering vector $\underline{s}_\theta$. Thus, the beamformer response along a direction characterized by the angle θ is

$$y_\theta = \langle \underline{w}, \underline{s}_\theta \rangle = \sum_{k=0}^{N-1} w_k \exp(jk_1 kD\sin\theta). \tag{12}$$

The beamforming system is said to be robust if it performs satisfactorily despite certain perturbations [Ahmed and Evans, 1982]. It is possible for each component $s_{k\theta}$ of $\underline{s}_\theta$ to belong to an interval $[s_{k\theta} - \phi_{k\theta}, s_{k\theta} + \phi_{k\theta}]$, and a robust beamformer will require the existence of at least one weight vector $\underline{w}$ which

will guarantee the output y_θ to belong to an output envelope for each s_θ in the input envelope. The robust beamforming problem can be translated into an optimization problem, which may be tackled by minimizing the value of the array output power

$$P(\theta) = \underline{w}^t(\theta) R \underline{w}(\theta) \tag{13}$$

when the response to a unit amplitude plane wave incident at the steering direction θ is constrained to be unity, i.e. $\underline{w}^t(\theta)\underline{s}(\theta) = 1$, and R is the additive noise corrupted signal autocorrelation matrix. The solution is called the minimum variance beamformer and is given by

$$\underline{w}_{MV}(\theta) = \frac{R^{-1}\underline{s}(\theta)}{\underline{s}^t(\theta) R^{-1} \underline{s}(\theta)}, \tag{14}$$

and the corresponding power output is

$$P_{MV}(\theta) = \frac{1}{\underline{s}^t(\theta) R^{-1} \underline{s}(\theta)}. \tag{15}$$

The minimum variance power as a function of θ can be used as a form of the data-adaptive estimate of the directional power spectrum. However, in this mode of solution, the coefficient vector is unconstrained except at the steering direction. Consequently, a signal tends to be regarded as an unwanted interference and is, therefore, suppressed in the beamformed output unless it is almost exactly aligned with the steering direction. Therefore, it is desirable to broaden the signal acceptance angle while at the same time preserving the optimum beamformer's ability to reject noise and interference outside this region of angles. One way of achieving this is by the application of the principle of superdirectivity.

Discrete Arrays and Polynomials:

It is common practice to relate discrete arrays to polynomials for array synthesis purposes [Steinberg, 1976]. For volumetric equispaced arrays (it is only necessary that the spacing be uniform along each of the coordinate axis so that the spatial sampling periods D_i and D_j along, respectively, the i^{th} and j^{th} coordinate axes could be different for $i \neq j$) the weight associated with sensors located at coordinate $(i_1 D_1, i_2 D_2, i_3 D_3)$ is denoted by $w(i_1, i_2, i_3)$. The function in the complex variables $z_1, z_2,$, and z_3 that is associated with the sequence $\{w(i_1, i_2, i_3)\}$ is the generating function for the sequence and is denoted by

$$W(z_1, z_2, z_3) = \sum_{i_1} \sum_{i_2} \sum_{i_3} w(i_1, i_2, i_3)\, z_1^{i_1} z_2^{i_2} z_3^{i_3}. \tag{16}$$

In the electrical engineering and in the geophysics literature, the generating function $W(z_1, z_2, z_3)$ is, sometimes, called the z-transform of the sequence $\{w(i_1, i_2, i_3)\}$. When there are a finite number of sensors, a realistic assumption for any physical discrete array, $W(z_1, z_2, z_3)$ becomes a trivariate polynomial. In the special case when $w(i_1, i_2, i_3)$ is product separable, the polynomial $W(z_1, z_2, z_3)$ is also product separable. Particularly, this separability property holds when the shading is uniform, i.e. $w(i_1, i_2, i_3) = 1$. When the support of the uniform shading function is defined by $i_1 = 0, 1, \ldots, N_1 - 1$, $i_2 = 0, 1, \ldots, N_2 - 1$, and $i_3 = 0, 1, \ldots, N_3 - 1$, the associated polynomial becomes

$$W(z_1, z_2, z_3) = \sum_{i_1=0}^{N_1-1} \sum_{i_2=0}^{N_2-1} \sum_{i_3=0}^{N_3-1} z_1^{i_1} z_2^{i_2} z_3^{i_3} = \prod_{i=1}^{3} \frac{z_i^{N_i}-1}{z_i-1} \tag{17}$$

In this case, all results developed for the synthesis of linear arrays become directly applicable to the synthesis of volumetric arrays. For a linear uniform discrete array composed of N sensors with intersensor spacing D_1 starting at the origin and receiving a signal at a known fixed wavenumber k_1 at a receiving angle θ, the far-field beampattern

$$S(k_1, \theta) \triangleq S(\theta) = \sum_{r=0}^{N-1} e^{jk_1 r D_1 \sin\theta}$$

may be associated with a polynomial $\sum_{r=0}^{N-1} z_1^r$, by setting $z_1 = e^{jk_1 D_1 \sin\theta}$. This polynomial has all its zeros on the unit circle in the z_1-plane. If the array just considered is not uniform but has a weighting factor w_r, for $r = 0,1,\ldots,N_1-1$, the space factor,

$$Q(\theta) \triangleq \sum_{r=0}^{N_1-1} w_r e^{jk_1 D_1 r \sin\theta}$$

may again be associated with a polynomial $\sum_{r=0}^{N_1-1} w_r z_1^r$. By the pattern multiplication theorem, it is possible to get the polynomial associated with the total beam pattern of an array with weighted sensors by multiplying the polynomials associated with the array element pattern and the polynomial associated with the space factor $Q(\theta)$. The array factor $|Q(\theta)|^2$ may also be associated with the polynomial spectral factor

$$|Q(\theta)|^2 \leftrightarrow \sum_{r=0}^{N_1-1} w_r z_1^r \sum_{r=0}^{N_1-1} w_r^* (z_1^*)^r, \tag{18}$$

where the weighting (shading) factor is allowed to be complex. Uniformly distributed apertures and uniformly spaced volumetric arrays which admit product separable sensor weightings can be treated by using the well-developed theory of linear discrete arrays and their associated polynomial. When the product separability property does not hold, scopes exist for applying results from multidimensional systems theory [Bose, 1982] concerning multivariate polynomials to the synthesis problem of volumetric arrays.

Velocity Filtering:

Combination of individual sensor outputs in a more sophisticated way than delay-and-sum technique leads to the design of multichannel velocity filters for linear, planar, as well as spatial arrays. Consider, first, a linear (1-D) array of sensors, which will be used to implement velocity discrimination. The pass and rejection zones are defined by straight lines in the (k_1, ω)-plane, where

$$k_1 = \frac{\omega}{V} = \frac{\omega}{(v/\sin\theta)}$$

is the wavenumber, ω the angular frequency in radians/sec, V the apparent velocity on the earth's surface along the array line, v is the velocity of wave propagation, and θ is the horizontal arrival direction. The transfer

function

$$H(\omega, k_1) = \begin{cases} 1, & -\frac{|\omega|}{V} \le k_1 \le \frac{|\omega|}{V} \\ 0, & otherwise, \end{cases}$$

of a "pie-slice" or "fan" velocity filter [Bose, 1985] rejects totally wavenumbers outside the range, $-\frac{|\omega|}{V} \le k_1 \le \frac{|\omega|}{V}$, and passes completely wavenumbers defined within that range. Thus, the transfer function defines a high-pass filter which passes signals with apparent velocities of magnitude greater than V at a fixed frequency ω. If the equi-spaced sensors are D units apart, the spatial sampling results in a periodic wavenumber response with period $k_1 = \frac{1}{2D}$. Therefore, for a specified apparent velocity V, the resolvable wavenumber and frequency bands are, respectively, $-\frac{1}{2D} \le k_1 \le \frac{1}{2D}$ and $-\frac{V}{2D} \le \omega \le \frac{V}{2D}$ where $\frac{\omega}{2D}$ represents the folding frequency in radians/sec.

Linear arrays are subject to the limitation that the source is required to be located on the extended line of sensors so that plane wavefronts approaching the array site at a particular velocity excite the individual sensors, assumed equi-spaced, at arrival times which are also equi-spaced. In seismology, the equi-spaced interval between successive sensor arrival times is called a move-out or step-out and equals $\frac{D\sin\theta}{V} = \frac{D}{V}$. However, when the sensor-to-source azimuth varies, two or more independent signal move-outs may be present. Planar (2-D) arrays are then required to discriminate between velocities as well as azimuth. Spatial (3-D) arrays provide additional scope to the enhancement of discriminating capabilities when sensor/source locations are arbitrary. In such cases, an array origin is chosen and the mth sensor location is denoted by a vector $(x_{1m}\ x_{2m}\ x_{3m})^t$ and the frequency wavenumber response of an array of N sensors is given by

$$H(\omega, k_1, k_2, k_3) = \frac{1}{N}\sum_{m=1}^{N} H_m(\omega) \exp\left[\sum_{i=1}^{3} -j2\pi k_i x_{im}\right]$$

where $H_m(\omega)$ denotes the frequency response of a filter associated with the mth recording device (sensor). The sum of all N filters provides flat frequency response so that waveforms arriving from the estimated directions of arrival at estimated velocities are passed undistorted and other waveforms are suppressed. In the planar specialization, the 2-D array of sensors lead to the theory of 3-D filtering involving a transfer function in the frequency/wavenumber variables f, k_1 and k_2. The basic design equations for the optimum, in the least-mean-square error sense, frequency filters have been developed [Burg, 1964]. This procedure of Burg can be routinely generalized to the 4-D filtering problem mentioned above.

Acknowledgement

N. K. Bose and L. H. Sibul acknowledge the supports provided by the Office of Naval Research under, respectively, Contract N0014-86-K-0542 and the Fundamental Research Initiatives Program.

References

Ahmed, K.M. and Evans, R.J., 1982. "Robust signal and array processing," IEE Proceedings F. Communications, Radar, and Signal Processing, 129, (4), pp. 297-302.

Bose, N.K., 1982. *Applied Multidimensional Systems Theory*, Van Nostrand Reinhold, New York.

Bose, N.K., 1985. *Digital Filters*, Elsevier Science North-Holland, New York.

Burg, J. P., 1964. "Three-dimensional filtering with an array of seismometers," *Geophysics*, 23, 5, pp. 693-713.

Claerbout, J.F., 1976. *Fundamentals of Geophysical Data Processing*, McGraw-Hill, New York.

Dudgeon, D.E., 1977. "Fundamentals of digital array processing," *Proceedings of IEEE*, 65. pp. 898-904.

Monzingo, R.A. and Miller, T.W., 1980. *Introduction to Adaptive Arrays*, John Wiley & Sons, Inc., New York.

Pillai, S.M. 1989. *Array Signal Processing*, Springer-Verlag, New York.

Steinberg, B.D., 1976. *Principles of Aperture and Array System Design*, John Wiley & Sons, Inc., New York.

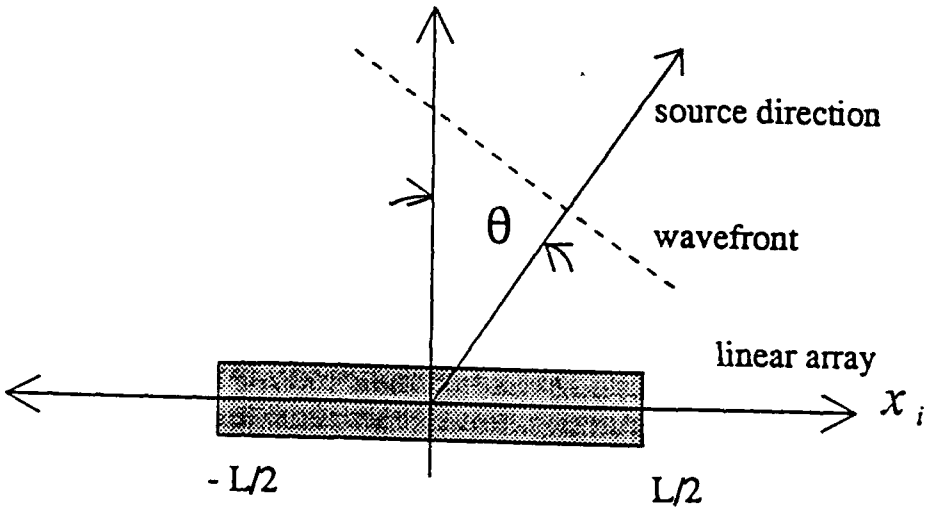


Figure 1. Uniformly weighted linear array.

A Comprehensive Analysis of "Neural Solution to the Multitarget Tracking Data Association Problem" *

B. Zhou and N. K. Bose

Department of Electrical and Computer Engineering
The Spatial and Temporal Signal Processing Center
Pennsylvania State University
University Park, PA16802
(814)-865-3912

Abstract

A comprehensive analysis reveals several drawbacks of the neural network probabilistic data association (NPDA) algorithm, which is an application of the Hopfield neural network to the data association problem for multitarget tracking in clutter.

*This research has been sponsored by SDIO/IST and managed by the Office of Naval Research under contract N00014-86-K-0542

1 Introduction

Sengupta and Iltis [1] formulated the computation of the *a posteriori* probabilities for the joint probabilistic data association filter (JPDAF) [2] as a constrained minimization problem. This technique based on the use of neural networks was also extended [3] to apply to maneuvering targets. The *a posteriori* probability β_j^t (for $j \neq 0$) is the probability that measurement j originated from target t and β_0^t is the probability that none of the received measurements originated from target t. By comparison with the traveling salesman problem (TSP), they proposed a Hopfield neural network [4] to approximately compute the β_j^t's and called this neural network probabilistic data association (NPDA). In fact, β_j^t is approximated by the output voltage V_j^t of a neuron in an $(m+1) \times n$ array of neurons, where m is the number of measurements and n is the number of targets. Sengupta and Iltis claimed that the performance of the NPDA was close to that of the JPDAF in situations where the numbers of measurements and targets were in the ranges of 3 to 20 and 2 to 6, respectively. The success of the NPDA in their examples was credited to the accurate emulation of all the properties of the JPDAF by the NPDA.

Since there are no strict guidelines for choosing the constant coefficients of the energy function for the Hopfield neural networks [4], these coefficients in the NPDA were selected essentially by trial-and-error [1]. However, with the chosen coefficients in [1], the architecture of the neural network in [1] degenerates into individual columns with identical connections and the input currents throughout the neural network are almost of the same amplitude. Furthermore, the neural network in [1] has a strong tendency to converge to the V_j^t's which are close to $1/(m+1)$ with the given initial values for the V_j^t's. In [3], a least-squares approach was developed to optimize the last two coefficients of the energy function. The authors pointed out that this local optimization of the two coefficients is quite sensitive to n, the number of targets. They chose the two coefficients by averaging their values for the $n = 2$, $n = 4$, and $n = 6$ cases. As a result, the coefficients of the energy function in [3] are slightly different from the ones in the energy function of [1]. Because of the similarity between the energy functions in [1, 3], the discussion in this correspondence will be focused on the

energy function in [1]. Similar discussions on the energy function in [3] are documented in detail in [5].

In Section 2, the Hopfield neural network used in [1] is briefly reviewed and some comments are made on the assumptions which were used to set up the energy function in [1]. In Section 3, some criticisms on the implementation of the neural network in [1] are given. Conclusions are summarized in Section 4.

2 Review of the Energy Function in the NPDA and Comments

Suppose there are n targets and m measurements. The energy function used in [1] is reproduced below.

$$\begin{aligned} E_{DAP} &= \frac{A}{2}\sum_{j=0}^{m}\sum_{t=1}^{n}\sum_{\substack{\tau=1\\ \tau\neq t}}^{n} V_j^t V_j^\tau + \frac{B}{2}\sum_{t=1}^{n}\sum_{j=0}^{m}\sum_{\substack{l=0\\ l\neq j}}^{m} V_j^t V_l^t + \frac{C}{2}\sum_{t=1}^{n}(\sum_{j=0}^{m} V_j^t - 1)^2 + \\ &\quad \frac{D}{2}\sum_{j=0}^{m}\sum_{t=1}^{n}(V_j^t - \rho_j^t)^2 + \frac{E}{2}\sum_{j=0}^{m}\sum_{t=1}^{n}\sum_{\substack{\tau=1\\ \tau\neq t}}^{n}(V_j^t - \sum_{\substack{l=0\\ l\neq j}}^{m}\rho_l^\tau)^2. \end{aligned} \tag{1}$$

In (1), V_j^t is the output voltage of a neuron in an $(m+1)\times n$ array of neurons and is the approximation to the *a posteriori* probability β_j^t in the JPDAF [2]. This *a posteriori* probability, in the special case of the PDAF [2] when the probability P_G that the correct measurement falls inside the validation gate is unity, is denoted by ρ_j^t. Actually, P_G is very close to unity when the validation gate size is adequate, as shown in Table 5-1 in [2]. In (1), A, B, C, D, and E are constants.

In the NPDA, the connection strength matrix is a symmetric matrix of order $n(m+1)$. With the given energy function E_{DAP} in (1), the connection strength $T_{jl}^{t\tau}$ from the neuron at location (τ, l) to the neuron at location (t, j) is

$$T_{jl}^{t\tau} = \begin{cases} -(C+D+E(n-1)), & \text{if } t=\tau \text{ and } j=l \quad \text{"self feedback"},\\ -A, & \text{if } t\neq\tau \text{ and } j=l \quad \text{"row connection"},\\ -(B+C), & \text{if } t=\tau \text{ and } j\neq l \quad \text{"column connection"},\\ 0, & \text{if } t\neq\tau \text{ and } j\neq l \quad \text{"global connection"}. \end{cases} \tag{2}$$

The input current I_j^t to the neuron at location (t, j), for $t = 1, 2, \ldots, n$, and $j = 0, 1, \ldots, m$, is

$$I_j^t = C + (D + E)\rho_j^t + E(n - 1 - \sum_{\tau=1}^{n} \rho_j^\tau). \tag{3}$$

Clearly from (2) and (3), the input current I_j^t but not the connection strength $T_{jl}^{t\tau}$ depends on the ρ_j^t's, which are computed from the measurements that comprise the input data. Ironically, in the neural network for the TSP [4], only the connection strengths depend on the input data which, in this case, are the distances between pairs of cities.

In order to justify the first two terms of E_{DAP} in (1), the authors of [1] claimed that the dual assumptions of no two returns from the same target and no single return from two targets are consistent with the presence of a dominating V_j^t in each row and each column of the $(m + 1) \times n$ array of neurons. However, these assumptions are not constraints on the values of the β_j^t's in the original JPDAF. Those assumptions should be used only in the generation of the feasible data association hypotheses, as pointed out in [2, p. 224]. As a matter of fact, there could be two β_j^t's of comparable magnitude in the same row and in the same column as shown in chapter 4 of [5]. Therefore, the presence of a dominating V_j^t in each row and each column is not a property of the JPDAF.

The third term of E_{DAP} is used to constrain the sum of the V_j^t's in each column to unity *i.e.* $\sum_{j=0}^{m} V_j^t = 1$. This constraint is consistent with the requirement that $\sum_{j=0}^{m} \beta_j^t = 1$ in both the JPDAF and the PDAF [2]. Therefore, this constraint, by itself, does not permit us to infer whether the β_j^t's are from the JPDAF, or from the PDAF. The assumption used to set up the fourth term is that this term is small only if V_j^t is close to ρ_j^t, in which case the neural network simulates more closely the PDAF for each target rather than the intended JPDAF in the multitarget scenario. Finally, the fifth term is supposed to be minimized if V_j^t is not large unless for each $\tau \neq t$ there is a unique $l \neq j$ such that ρ_l^τ is large. Unfortunately, this constrained minimization may not be possible as shown in [5]. This is consistent with the heuristic nature of the derivation of the energy function in [1], which could lead to the problems in the implementation of the NPDA as discussed next.

3 Criticism of the Implementation of the NPDA

Since there are no guidelines for choosing A, B, C, D, and E in (1), these coefficients are usually set through simulation. In the implementation of the NPDA [1], the simulations provided the following set of values for A, B, C, D, and E for tracking two to six targets with three to twenty measurements.

$$A = 0 \quad B = 40 \quad C = 1000$$
$$D = 30 \quad E = 10$$

After substituting the above set of values into (2), the only connections between neurons which are nonzero are those due to "self feedback" and "column connection", as evident from (4) below.

$$T_{jl}^{t\tau} = \begin{cases} -(1030 + 10(n-1)), & \text{if } t = \tau \text{ and } j = l \text{ "self feedback"} \\ 0, & \text{if } t \neq \tau \text{ and } j = l \text{ "row connection"} \\ -1040, & \text{if } t = \tau \text{ and } j \neq l \text{ "column connection"} \\ 0, & \text{if } t \neq \tau \text{ and } j \neq l \text{ "global connection".} \end{cases} \tag{4}$$

Since $0 \leq \rho_j^t \leq 1$, the upper and lower bounds of the input current I_j^t in (3) can be obtained.

$$1000 \leq I_j^t \leq 1030 + 10(n-1). \tag{5}$$

The upper bounds of the input current I_j^t in [1] are 1040, 1060, and 1080 for $n = 2, 4, 6$, respectively. Clearly, the maximum range of variation of the input current I_j^t is only a small percentage of the value of C. Therefore, the input current I_j^t is dominated by C for $n = 2, 4, 6$. In other words, the ρ_j^t's do not have much influence on the solution obtained from the neural network.

One may argue that C does not dominate the behavior of the neural networks if the dynamic equation (32) in [1] is considered. For the purpose of the discussion, this equation with $A = 0$ is reproduced below.

$$\frac{du_j^t}{ds} = -\frac{u_j^t}{s_0} - B\sum_{\substack{l=0 \\ l\neq j}}^{m} V_l^t - C\left(\sum_{l=0}^{m} V_l^t - 1\right) - (D + E(n-1))V_j^t$$

$$+(D+E)\rho_j^t + E(n-1-\sum_{\tau=1}^{n}\rho_j^\tau). \tag{6}$$

In (6), s_0 is the time constant of the neural network and u_j^t is the input voltage of the neuron located at (t,j) in the array of neurons. When the neural network is initialized so that each neuron approximately outputs

$$V_j^t = \frac{1}{m+1}, \tag{7}$$

the third term is almost zero because $\sum_{l=0}^{m} V_l^t$ is close to unity. However, we must keep in mind that this neural network has not been developed for implementation on a digital computer. Instead, it has been developed for possible implementation using an analog network. If it is implemented on a digital computer, (6) is replaced by (38) in [1], which is reproduced below for ready reference.

$$u_j^t(i+1) = \frac{s_0-\zeta}{s_0}u_j^t(i) - \zeta B\sum_{\substack{l=0 \\ l\neq j}}^{m} V_l^t - \zeta C\left(\sum_{l=0}^{m} V_l^t - 1\right) - \zeta(D+E(n-1))V_j^t$$

$$+\zeta(D+E)\rho_j^t + \zeta E(n-1-\sum_{\tau=1}^{n}\rho_j^\tau), \tag{8}$$

In (8), ζ is the iteration step and 12 multiplications are sufficient for implementing each iteration. A total of 400 iterations was used to guarantee the convergence of the neural network to a satisfactory solution [1]. There are altogether $n(m+1)$ equations like (8) for tracking n targets with m measurements. Therefore, the total number of multiplications required to compute the β_j^t's does not exceed $n(m+1)*12*400$. Comparing this number with the entries in Tables 2.4, 3.3, and 3.4 in [5], we can conclude that the implementation of the neural network on a computer is far more computationally expensive than the direct computation of the β_j^t's. If this neural network is implemented on an analog circuit, the dynamic equation of the network is

$$\frac{du_j^t}{ds} = -\frac{u_j^t}{s_0} - \sum_{\tau=1}^{n}\sum_{l=0}^{m} T_{jl}^{t\tau}V_l^\tau + I_j^t, \tag{9}$$

as shown in (3) of [4]. Therefore, in this example because of the I_j^t term in (9), it is seen from (5) that C becomes a dominan.. ..ctor in the neural network. Since the range of variation of I_j^t is only a small percentage of C, any perturbation in the implementation of the current source due to noise would contribute to the output voltage V_j^t, which is supposed to approximate β_j^t. Since C is significantly larger than A, B, D, and E, the network has a strong tendency to be locked into a state where $\sum_{l=0}^{m} V_l^t = 1$. With the above initialization, identical connections, and nearly uniform input currents, it is even easier for the network to converge to the normalized V_j^t's which are close to $1/(m+1)$. Therefore, it is possible for some V_j^t's to be not 0 even if the β_j^t's are 0. As a result, the state of target t is updated in part by measurement j which is not inside the validation gate of target t.

Finally, it is worth mentioning that the example given in [1] for tracking six targets is not actually a six-target-tracking problem from the JPDAF point of view. In multitarget tracking, targets are grouped into clusters before applying either the JPDAF or the PDAF. Targets are in the same cluster if there is at least one measurement in each of the intersections of the validation gates of these targets. The JPDAF is only applied to a cluster which contains more than one target. Otherwise, the PDAF is applied. If the targets in the examples given in [1] are grouped into clusters, the largest size of a cluster of targets is 3. In such cases, the β_j^t's can be easily computed by the algorithms proposed in chapters 2 and 3 of [5] and also reported in [6].

4 Summary

It might be a good idea to use the neural network to approximately compute the *a posteriori* probability β_j^t in the JPDAF. However, the neural network developed in [1] has been shown to have improper energy functions. This resulted from misinterpretations of the properties of the JPDAF which the network was supposed to emulate. Furthermore, improper choices of the constant coefficients in the energy function in [1] make the situation worse. Therefore, further study of the properties of the JPDAF is needed to construct a better energy function for the NPDA.

References

[1] D. Sengupta and R. A. Iltis, "Neural solution to the multiple target tracking data association problem," *IEEE Trans. Aerospace and Electronic Systems*, vol. AES-25, pp. 96–108, January 1989.

[2] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association.* Orlando, FL: Academic Press, Inc., 1988.

[3] D. Sengupta and R. A. Iltis, "Multiple maneuvering target tracking using neural networks," Tech. Rep. 90-32, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, December 1990.

[4] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[5] B. Zhou, *Multitarget Tracking in Clutter: Algorithms for Data Association and State Estimation.* PhD thesis, Pennsylvania State University, Department of Electrical and Computer Engineering, University Park, PA 16802, May 1992.

[6] B. Zhou and N. K. Bose, "Multitarget tracking in clutter: Fast algorithms for data association." *IEEE Trans. Aerospace and Electronic Systems,* submitted for publication.

Multitarget Tracking In Clutter: Fast Algorithms For Data Association*

B. Zhou and N. K. Bose

Department of Electrical Engineering

The Spatial and Temporal Signal Processing Center

Pennsylvania State University

University Park, PA16802

(814)-865-3912

Abstract

In this paper, three fast algorithms have been developed to solve the problem of data association in multitarget tracking in clutter. In the first algorithm, the problem of data association is identified as an *exhaustive search problem* in general. Subsequently, a mathematical model is proposed for the problem of data association in the joint probabilistic data association filter (JPDAF). Based on the model, a depth-first search (DFS) approach is developed for the fast generation of data association hypotheses and the computation of the conditional probabilities of the hypotheses in the JPDAF. With proper preprocessing, the number of multiplications in the computation of the conditional probabilities is less than twice the number of data association hypotheses. Therefore, the computational complexity of the algorithm is analyzed in terms of the number of data association hypotheses in the worst case and also in the average case. When the density of targets is moderate, an even more efficient algorithm is developed to directly compute *a posteriori* probabilities in the JPDAF without generating the data association hypotheses. In the third algorithm, the interference among closely spaced targets is simplified. This lead us to develop an approach to approximately compute the *a posteriori* probabilities in the JPDAF.

*This research has been sponsored by SDIO/IST and managed by the Office of Naval Research under contract N00014-86-K-0542

1 Introduction

In multitarget tracking in clutter, often there are more than one measurement available for updating the state of a single target. To solve the problem of data association between targets and measurements, two typical approaches have been reported in the literature in the 70's. One is called the *track-oriented* approach in which each measurement is assumed to have originated from either a known target or clutter, as in *Probabilistic Data Association Filter* (PDAF) [1,2] and *Joint Probabilistic Data Association Filter* (JPDAF) [1,3]. The other is called a *measurement-oriented* approach in which each measurement is hypothesized to have originated from either a known target, a new target, or clutter [4]. In both approaches, the number of data association hypotheses could increase rapidly with the increase of the number of targets and the number of measurements. Therefore, the computational cost in generating the data association hypotheses would overwhelmingly dominate in a multitarget tracking algorithm when the number of targets and the number of measurements are relatively large. In the PDAF, the computational cost for data association is reduced drastically by isolating targets from each other. Only the measurements which lie in the validation gate of a target are considered in data association. A validation gate is a specific region around the predicted position of a target. Since the PDAF ignores the interference from other targets, it may, sometimes, cause mistracking of closely spaced targets, as discussed by Fortmann, Bar-Shalom and Scheffe [3]. This difficulty is greatly reduced by using the JPDAF [3], which accounts for the fact that a measurement which falls inside the intersection of the validation gates of several targets could have originated from any one of these targets or from clutter. In the JPDAF, targets are divided into clusters. The targets are in the same cluster if there is at least one measurement inside each of the intersections of their validation gates. The computational cost for data association is reduced by grouping targets into clusters. The number of different data association hypotheses in each cluster is still an exponential function of the number of targets in the cluster. Due to the complexity of the problem of data association, the implementation

of a multitarget tracking algorithm has only been carried out in a 2-target case [3,4], except in [5]. In [5], an initial version of the depth-first search (DFS) algorithm proposed in this paper was used to track 11 non-maneuvering and maneuvering targets through the implementation of JPDA filtering and smoothing. Details of the proposed algorithm were not discussed previously.

To reduce significantly the computational cost for data association, several approximations of the JPDAF and the algorithm in [4] have been reported in literature. A first version of the approximate, or "cheap", JPDAF was published in [6], where the probability of association of target t with measurement j was computed by an *ad hoc* formula. The author of [6] developed his approach further in [7]. In [7], the performance of various versions of the approximate JPDAF's were tested. The "cheap JPDAF" in [6] performed fairly well in case of two targets but failed to track four targets [8]. In [8], the JPDAF was emulated by a neural network which is capable of handling two to six targets and three to twenty measurements with a set of carefully chosen coefficients for the network. In the neural network approach, the *a posteriori* probabilities, β_j^t's, of association of target t with measurement j were computed in parallel subject to the assumption that one out of all β_j^t's for any particular target t is significantly larger than the remaining β_j^t's for the same target. With this assumption, the resulting approximation of JPDAF is transformed more or less into the nearest-neighbor JPDAF [7]. Furthermore, in the neural network approach, targets are not grouped into clusters before calculating β_j^t as proposed in the original JPDAF [3]. Alternatively, Nagarajan, *et al.* [9] arranged the hypotheses in the measurement-oriented approach [4] in a special order so that the probabilities of the hypotheses are proportional to the product of certain probability factors already evaluated. The algorithm locates the N globally best hypotheses without evaluating all of them. For tracking multiple maneuvering targets in clutter, the authors of [8] introduced the approach of joint probabilistic data and maneuver association in [10].

In this paper, three fast algorithms have been developed. In the first algorithm,

a mathematical model is proposed for the problem of data association in general. This model is based on *exhaustive search*. Specifically, this model is tailored to the problem of data association in the JPDAF. Each feasible data association hypothesis in the JPDAF is equivalent to a solution to the exhaustive search problem for data association. Based on the proposed model, a depth-first search (DFS) algorithm is developed to generate data association hypotheses efficiently for the JPDAF. The manner in which the data association hypotheses are generated by the DFS algorithm suggests an approach to further reduce the number of multiplications in the computation of the conditional probabilities of the data association hypotheses. In fact, after proper preprocessing, the total number of multiplications in the computation of the conditional probabilities for the data association hypotheses in the JPDAF has been found to be less than twice the number of data association hypotheses generated by exhaustive search. The computational complexity of the proposed DFS algorithm is evaluated in terms of the number of hypotheses both in the worst and the average cases. The proposed DFS algorithm is most suitable for ground-based surveillance system with a large centralized computational capability, where accurate result is desired.

However, in a small tracking system or a missile guidance system where computational capability is limited, an even more faster algorithm is desired. The second and the third algorithms proposed in this paper are designated to be implemented in a small system or a system with multiprocessor. In the second algorithm, the *a posteriori* probabilities, β_j^t's, are computed directly without generating the data association hypotheses in the JPDAF when either the density of targets is not very high, or the size of the largest cluster of targets is less than or equal to 4. For the case in which the size of cluster of targets could be larger than 4, an approximation of the second algorithm is developed. In the approximation, only the interference from the neighboring targets of target t are considered in the computation of β_j^t. A target is a neighbor of target t if there is at least a measurement inside the intersection of the validation gates of the two targets. The second and the third algorithms are equivalent when the size of a cluster

of targets is 2. The important feature of the third algorithm is that β_j^t for each target is computed independently as in the PDAF algorithm.

A description of the problem of interest is provided in Section 2. In this section, the JPDAF is briefly reviewed to explain the motivation behind this research. Furthermore, an alternate formula for the conditional probabilities of the data association hypotheses is derived in order to simplify the notation in the development of the three algorithms. In Section 3.1, a comparison between a general exhaustive search problem and the one of data association in multitarget tracking is discussed. This leads to a mathematical model for the problem of data association. In Section 3.2 some unique features of the problem of data association are pointed out as a prelude to a specialized DFS algorithm for generating all data association hypotheses. In Section 3.3, the advantage of the DFS algorithm is further demonstrated in the computation of the conditional probabilities of the data association hypotheses. In Section 3.4, the computational complexity of the proposed DFS algorithm is analyzed. In Section 4, an algorithm is proposed to compute β_j^t directly when the density of targets is not very high. An approximation of the direct computation algorithm is presented in Section 5. Finally, in Section 6, computer simulation is carried out for tracking multitarget in different scenario.

2 Problem Formulation

Let m and n be the numbers of measurements and targets, respectively, in a particular cluster. β_0^t is the *a posteriori* probability for no measurement to originate from target t and β_j^t $(j \neq 0)$ is the *a posteriori* probability for measurement j to originate from target t. In the JPDAF, the computation of β_j^t's begins with the construction of a so-called validation matrix Ω for the n targets and m measurements. The validation matrix Ω is a $m \times (n+1)$ rectangle matrix defined as [3]

$$\Omega = [\omega_{jt}] = \begin{array}{c} \overbrace{}^{t} \\ \begin{array}{ccccc} 0 & 1 & 2 & \cdots & n \end{array} \\ \left(\begin{array}{ccccc} 1 & \omega_{11} & \omega_{12} & \cdots & \omega_{1n} \\ 1 & \omega_{21} & \omega_{22} & \cdots & \omega_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_{m1} & \omega_{m2} & \cdots & \omega_{mn} \end{array}\right) \begin{array}{c} 1 \\ 2 \\ \vdots \\ m \end{array} \end{array} \Bigg\} j. \qquad (1)$$

where $\omega_{j0} = 1$, $\omega_{jt} = 1$ if measurement j is inside the validation gate of target t and $\omega_{jt} = 0$ if measurement j is outside the validation gate of target t for $j = 1, 2, \ldots, m$, and $t = 1, 2, \ldots, n$. Based on the validation matrix Ω, data association hypotheses (or feasible events [3]) are generated subject to the following two restrictions:

1. each measurement can have only one origin (either a specific target or clutter).

2. no more than one measurement originates from a target.

This leads to a combinatorial problem where the number of data association hypotheses increases exponentially with the number of targets and the number of measurements.

Each feasible event $\mathcal{E}$ is represented by a hypothesis matrix $\hat{\Omega}$ in [3]. $\hat{\Omega}$ has the same size as the validation matrix Ω. A typical element in $\hat{\Omega}$ is denoted by $\hat{\omega}_{jt}$ where $\hat{\omega}_{jt} = 1$ only if measurement j is hypothesized to be associated with clutter $(t = 0)$ or target t $(t \neq 0)$. After each $\hat{\Omega}$ is obtained, the conditional probability of the corresponding

data association hypothesis or feasible event is calculated by a formula given in [3]. A simplified version of this formula is given in (2), where time index k is omitted.

$$P(\mathcal{E}(\hat{\Omega})|Z) = \frac{1}{c}(P_0)^{\min(n,m)-m_a} \prod_{j:\hat{\omega}_{jt}=1} P_j^t \tag{2}$$
$$\text{for } j = 1,2,\ldots,m \text{ and } t = 1,2,\ldots,n.$$

where Z is the set of all measurements received up to current time index k, c is a normalizing constant, m_a is the number of targets detected in this feasible event $\mathcal{E}$, $\hat{\omega}_{jt} = 1$ indicates that measurement j is associated with target t in the event, and

$$P_j^t = \begin{cases} N(\tilde{z}_j^t; 0, S^t) P_D & \text{if } \omega_{jt} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$P_0^t = \lambda(1 - P_D) = P_0. \tag{4}$$

$$\text{for } j = 1,2,\ldots,m \text{ and } t = 1,2,\ldots,n.$$

In (3) and (4), λ is the clutter density, P_D is the probability of detection, and $N(\tilde{z}_j^t; 0, S^t)$ is a normal distribution density function with zero mean and covariance matrix S^t. It is understood that the normalizing constant c in (2) is obtained by the summation, $\sum_{\mathcal{E}(\hat{\Omega})} P(\mathcal{E}(\hat{\Omega})|Z)$. Therefore, c is omitted hereafter. The *a posteriori* probability β_j^t is computed from the conditional probabilities in (2) by

$$\beta_j^t = \sum_{\mathcal{E}(\hat{\Omega})} P(\mathcal{E}(\hat{\Omega})|Z)\hat{\omega}_{jt} \tag{5}$$

$$\beta_0^t = 1 - \sum_{j=1}^{m} \beta_j^t \tag{6}$$

$$\text{for } j = 1,2,\ldots,m, \text{ and } t = 1,2,\ldots,n.$$

In order to have a better understanding of how each β_j^t is calculated, consider an example. Suppose, there are two targets. In one radar scan, three measurements are received and suppose one of these falls inside the intersection of the validation gates of the two targets. The mathematical representation of this situation may be described by a validation matrix Ω:

$$\Omega = \begin{array}{c} \overbrace{}^{t} \\ \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \left.\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}\right\} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \end{array} \Bigg\} \text{j.} \tag{7}$$

With the two restrictions given above, 8 feasible events $\mathcal{E}(\hat{\Omega}_l)$ $(l = 0, 1, \ldots, 7)$ may be constructed. The hypothesis matrices are

$$\hat{\Omega}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \hat{\Omega}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \hat{\Omega}_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\hat{\Omega}_3 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \hat{\Omega}_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \hat{\Omega}_5 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\hat{\Omega}_6 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \hat{\Omega}_7 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The conditional probability $P(\mathcal{E}(\hat{\Omega}_l)|Z)$ can be easily calculated by using (2). Subsequently, all β_j^t's may be computed. For example,

$$\beta_1^1 = P(\mathcal{E}(\hat{\Omega}_1)|Z) + P(\mathcal{E}(\hat{\Omega}_2)|Z) + P(\mathcal{E}(\hat{\Omega}_3)|Z)$$

The problems of interest here in the implementation of the JPDAF for any number of targets and measurements are the efficient generation of hypothesis matrices $\hat{\Omega}_l$ and fast computation of β_j^t.

3 The Depth-First Search (DFS) algorithm

3.1 Mathematical Model for Data Association

As pointed out earlier, data association in multitarget tracking is a combinatorial problem. The computational cost of data association increases exponentially with n, the number of targets, and m, the number of measurements. The efficiency of the algorithm used in the generation of the data association hypotheses is especially important when n and m are relatively large. In order to develop an efficient algorithm to generate all data association hypotheses, it is necessary that its nature be well understood. So a mathematical model is developed for data association.

A well-known model for a combinatorial problem is called *exhaustive search with certain constraints*[11]. The general description of a typical exhaustive search problem is following:

> There are m variables X_j $(j = 1, 2, \ldots, m)$. The value of each X_j belongs to a set Z_j, where Z_j is finite and linearly ordered. A candidate solution for this problem is a m-tuple
>
> $$(X_1, X_2, \ldots, X_m).$$
>
> The objective here is to find either one solution or all solutions each of which satisfies the imposed constraints.

An efficient algorithm to solve the above problem is a depth-first search (DFS) procedure which involves backtracking [11]. In the DFS procedure, a solution is found by checking the m variables in a m-tuple one by one from left to right, as shown in Fig. 1. In Fig. 1, $GET\ NEXT(Z_j, X_j)$ is a logical function. It is true only if $X_j \in Z_j$, X_j has not been used in the DFS procedure yet, and X_1, X_2, ..., and X_j satisfy the given constraints. Otherwise, it is false.

In the context of tracking multitarget in clutter, however, data association can be modelled as an exhaustive search problem with a set of proper notations. Let X_j ($j = 1, 2, \ldots, m$) denote measurement j. The value of X_j identifies the target or clutter which is hypothesized to be associated with measurement j. For example, $X_j = 3$ implies that measurement j is hypothesized to be associated with target 3. However, if $X_j = 0$, measurement j is hypothesized to be associated with clutter. Z_j is, therefore, a set of clutter denoted by 0 and those targets for which $\omega_{jt} = 1$ in the validation matrix. Equivalently,

$$Z_j = \{t | \omega_{jt} = 1\}, \quad j = 1, 2, \ldots, m, \text{ and } t = 0, 1, 2, \ldots, n. \tag{8}$$

Note that since ω_{j0} is equal to 1, therefore $0 \in Z_j$ for $j = 1, 2, \ldots, m$. If measurement j falls inside the validation gate of target t, $t \in Z_j$. For the example discussed in the previous section, the Z_j's ($j = 1, 2, 3$) are:

$$Z_1 = \{0, 1\}; \quad Z_2 = \{0, 1, 2\}; \quad Z_3 = \{0, 2\}.$$

In the JPDAF scenario, the two constraints which have to be satisfied for a feasible event can be easily translated into the language of exhaustive search problem for data association. Hence, a m-tuple

$$(X_1, X_2, \ldots, X_p, \ldots, X_q, \ldots, X_m)$$

is a solution if the following two constraints are satisfied.

1. If $p \neq q$, $X_p \neq 0$, and $X_q \neq 0$, then $X_p \neq X_q$.

2. If $X_p = X_q$ and $p \neq q$, then $X_p = X_q = 0$.

All data association hypotheses can be generated by solving the exhaustive search problem described above. In the next section, a DFS algorithm will be discussed in detail for generating all data association hypotheses.

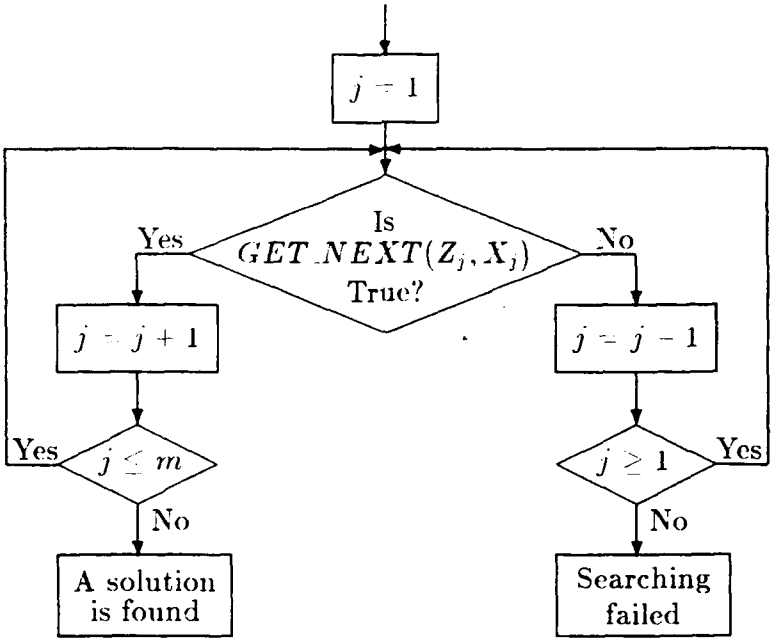


Figure 1 The flowchart of DFS procedure

3.2 Generation of Data Association Hypotheses

In the previous subsection, the similarities between the problem of exhaustive search and that of data association have been discussed and, subsequently, a mathematical model has been proposed for the problem of data association in multitarget tracking. It has also been pointed out that the DFS procedure is an efficient algorithm for finding solutions for the problem of exhaustive search. In this subsection, a specialized depth-first search DFS algorithm for the generation of data association hypotheses will be proposed. Before the DFS algorithm is given, it would be worthwhile to observe some differences between the problem of exhaustive search and that of data association.

Usually, there are no solutions that are known in advance in a general exhaustive search problem. However, in the problem of data association, a solution, which is always known in advance, is $(0,0,\ldots,0)$. The remaining solutions can be generated systematically from various valid combinations of non-zero values of the elements. In order to illustrate how this idea works, consider again the example in the previous sections. There are 2 targets and 3 measurements, and one of the 3 measurements is shared by the 2 targets. X_1, X_2, and X_3 denote the 3 measurements and the corresponding Z_j's are:

$$Z_1 = \{0,1\}; \quad Z_2 = \{0,1,2\}; \quad Z_3 = \{0,2\}.$$

The starting solution is (0,0,0). The next solution we look at is (1,0,0), obtained by setting $X_1 = 1$. Since 1 is already in the previous solution, the only non-zero value X_2 may have is 2. Therefore, after (1,0,0), the solution (1,2,0) is obtained. Now, all non-zero values in Z_2 which can been used to generate new solutions have been used up. Reset X_2 to 0 and go on to check if there is any non-zero value in Z_3 which will lead us to a new solution. In this case, by setting X_3 to 2, another solution, (1,0.2), is found. After all non-zero values in both Z_2 and Z_3 are used up, X_1 is reset to 0 and is kept as 0 for the remainder of the search process. At this stage, $X_2 = 1$ will not cause any conflict. Hence, a new solution, (0,1,0) is generated. This process can go on until all solutions are

found. In this particular example, the complete set of solutions is

$$\mathcal{E}_0 = (0,0,0), \quad \mathcal{E}_1 = (1,0,0), \quad \mathcal{E}_2 = (1,2,0),$$
$$\mathcal{E}_3 = (1,0,2), \quad \mathcal{E}_4 = (0,1,0), \quad \mathcal{E}_5 = (0,1,2),$$
$$\mathcal{E}_6 = (0,2,0), \quad \mathcal{E}_7 = (0,0,2).$$

For a better view of the solution generation process, all solutions can be arranged into a tree, which is called *hypotheses tree*, as shown in Fig. 2. Each node in the tree

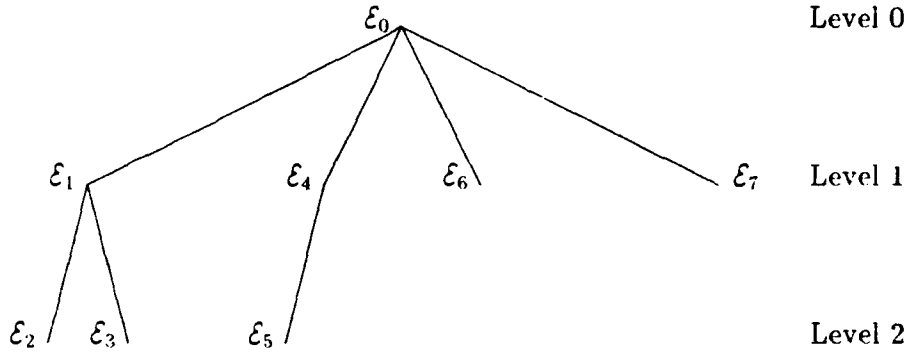


Figure 2 The graph arrangement of $\mathcal{E}_l$

stands for a solution or a data association hypothesis. The root of the tree is $\mathcal{E}_0$, which implies that all 3 measurements are hypothesized to be associated with clutter. Each node at level i is associated with i non-zero variables in the corresponding 3-tuple. This also implies that i out of 3 measurements are hypothesized to be associated with i out of 2 targets. In Fig. 2, the height of the tree is 2. In general, the height of the tree is $\min(n, m)$ since i must be less than both n and m.

As pointed out above, the number of non-zero variables in a 3-tuple indicates the level at which the corresponding node is located. However, the locations of the nodes at the same level are decided by which variables are non-zero and their values. Suppose that there are two 3-tuples $\mathcal{E}_x$ and $\mathcal{E}_y$ and that both of the two 3-tuples have 2 non-zero variables. In $\mathcal{E}_x$, variables X_{j_1} and X_{j_2} are non-zero and in $\mathcal{E}_y$, variables Y_{j_1} and Y_{j_2} are non-zero. $\mathcal{E}_x$ is to the left side of $\mathcal{E}_y$ if either

- $X_{j_1} = Y_{j_1^*}$ and $X_{j_2} = Y_{j_2^*}$ when $j_1 = j_1^*$ and $j_2 = j_2^*$, or
- $j_1 < j_1^*$ and $j_2 < j_2^*$.

For example, $\mathcal{E}_2$ $(= (1,2,0))$ and $\mathcal{E}_3$ $(= (1,0,2))$ have 2 non-zero variables. By letting $\mathcal{E}_2$ be $\mathcal{E}_x$ and $\mathcal{E}_3$ be $\mathcal{E}_y$, we have $X_1 = 1$, $X_2 = 2$, $Y_1 = 1$, and $Y_3 = 2$. Since $j_1 = j_1^* = 1$ and $2 = j_2 < j_2^* = 3$, $\mathcal{E}_2$ is at the left side of $\mathcal{E}_3$, as shown in Fig. 2. Therefore, the information about a data association hypothesis is contained in the non-zero variables in the 3-tuple. This is also true in the case of n targets and m measurements.

Because of the above distinguish features of the data association hypotheses, only non-zero variables in a hypothesis are stored in a stack of integer pairs in the implementation of the DFS algorithm. For example, $\mathcal{E}_4$ $(= (0,1,0))$ is represented by $((2,1))$. The first integer in the pair indicates which variable is non-zero in $\mathcal{E}_4$ and the second integer is the value of that variable. In general, any m-tuple with L non-zero variables is represented by

$$\vec{X} = \begin{pmatrix} (j_1, X_{j_1}) \\ (j_2, X_{j_2}) \\ \vdots \\ (j_L, X_{j_L}) \end{pmatrix} \tag{9}$$

where X_{j_1}, X_{j_2}, ..., and X_{j_L} are non-zero and L is the pointer of the stack $\vec{X}$ which is pointing at the last pair of integers in the stack. The flowchart of the DFS algorithm for the generation of the data association hypotheses is shown in Fig. 3. The proposed DFS algorithm is designated to visit every node in a hypothesis tree. Similar as in Fig. 1, $GET\ NEXT(Z_j, X_j)$ in Fig. 3 is a logical function. It is true only if $X_j \in Z_j$ and X_j is not equal to X_1, X_2, ..., and X_{j_L}. Otherwise, it is false.

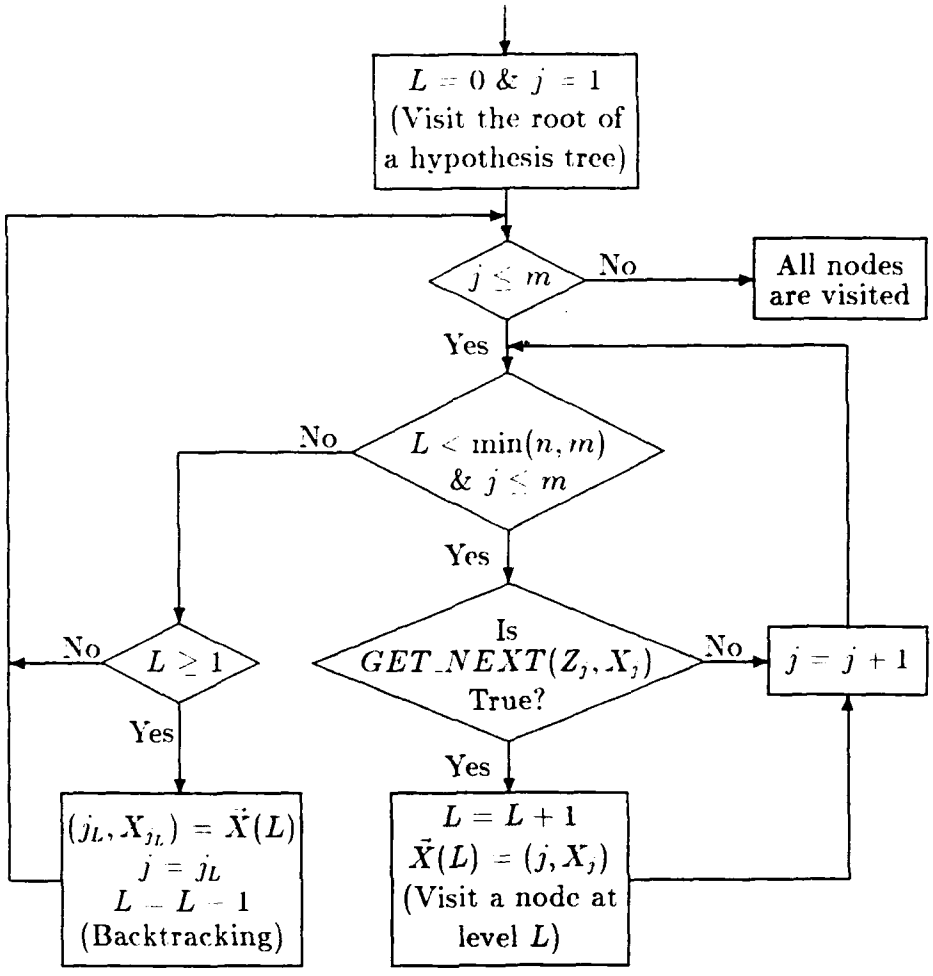


Figure 3 The flowchart of DFS procedure for data association

3.3 Computation of the Conditional Probabilities

After each data association hypothesis is generated, the conditional probability of the hypothesis is computed by using the formula in (2). However, if the data association hypotheses are arranged in such a order that a factor of a conditional probability is included in another conditional probability, the computation of the two conditional probabilities can be saved by sharing the common factor. The data association hypotheses generated by the proposed DFS algorithm are in such a special order that the common factors in the conditional probabilities can be shared in the most efficient way. If a little more complex example than the one considered in the previous sections, the advantage of the common factors in the computation of the conditional probabilities may be easily demonstrated. Suppose, there are 3 targets and 4 measurements, measurement 2 is inside the intersection of the validation gates of targets 1 and 2, and measurement 3 is inside the intersection of the validation gates of targets 2 and 3. Conventionally, this situation may be represented by a validation matrix Ω:

$$\Omega = \overbrace{\begin{matrix} 0 & 1 & 2 & 3 \end{matrix}}^{t} \quad \left.\begin{matrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \end{matrix}\right\} \text{j.} \tag{10}$$

However, it may also be represented by the notations introduced in the mathematical model, exhaustive search with constraints, proposed for data association. In the second representation, four variables, X_1, X_2, X_3, and X_4, are used to denote the 4 measurements and the corresponding Z_j's are:

$$Z_1 = \{0,1\}; \quad Z_2 = \{0,1,2\}; \quad Z_3 = \{0,2,3\}; \quad Z_4 = \{0,3\}.$$

When the DFS algorithm is applied to this example, the following sequence of solutions or data association hypotheses are generated

$$\mathcal{E}_0 = (0,0,0,0), \quad \mathcal{E}_1 = (1,0,0,0), \quad \mathcal{E}_2 = (1,2,0,0),$$
$$\mathcal{E}_3 = (1,2,3,0), \quad \mathcal{E}_4 = (1,2,0,3), \quad \mathcal{E}_5 = (1,0,2,0),$$
$$\mathcal{E}_6 = (1,0,2,3), \quad \mathcal{E}_7 = (1,0,3,0), \quad \mathcal{E}_8 = (1,0,0,3).$$

The partial hypothesis tree for the solutions listed above is shown in Fig. 4. Now, the conditional probabilities of the data association hypotheses $\mathcal{E}_l$ $(l = 0,1,\ldots,8)$ can be computed using (2). Recalling that Z denotes the set of all measurements received up to the current time index as defined on page 7.

$$P(\mathcal{E}_0|Z) = (P_0)^3, \quad P(\mathcal{E}_1|Z) = (P_0)^2 P_1^1, \quad P(\mathcal{E}_2|Z) = P_0 P_1^1 P_2^2,$$
$$P(\mathcal{E}_3|Z) = P_1^1 P_2^2 P_3^3, \quad P(\mathcal{E}_4|Z) = P_1^1 P_2^2 P_4^3, \quad P(\mathcal{E}_5|Z) = P_0 P_1^1 P_3^2,$$
$$P(\mathcal{E}_6|Z) = P_1^1 P_3^2 P_4^3, \quad P(\mathcal{E}_7|Z) = P_0 P_1^1 P_3^3, \quad P(\mathcal{E}_8|Z) = P_0 P_1^1 P_4^3.$$

It is not difficult to identify that P_1^1 is the common factor used in the computation of $P(\mathcal{E}_l|Z)$ $(l = 1,2,\ldots,8)$. However, the common factors, $P_1^1 P_2^2$ and $P_1^1 P_3^2$, need a closer examination. $P_1^1 P_2^2$ is used in the computation of $P(\mathcal{E}_l|Z)$ $(l = 2,3,4)$, while $P_1^1 P_3^2$ is used in the computation of $P(\mathcal{E}_5|Z)$ and $P(\mathcal{E}_6|Z)$.

If P_j^t's, P_0, $(P_0)^2$, and $(P_0)^3$ are computed in advance for this example and the three common factors mentioned above are used in the computation of $P(\mathcal{E}_l|Z)$ $(l = 0,1,\ldots,8)$, the number of multiplications required in the computation of $P(\mathcal{E}_l|Z)$'s may be obtained. For example, there is no multiplication required to obtain $P(\mathcal{E}_0|Z)$. To calculate $P(\mathcal{E}_1|Z)$, the product of $(P_0)^2$ and P_1^1 is required. In the computation of $P(\mathcal{E}_2|Z)$, the product of P_0, P_1^1, and P_2^2 is required. Therefore, $P(\mathcal{E}_1|Z)$ and $P(\mathcal{E}_2|Z)$ are computable with one and two multiplications. Since the product, $P_1^1 P_2^2$, is obtained in the computation of $P(\mathcal{E}_2|Z)$, there is only one multiplication required to compute of $P(\mathcal{E}_3|Z)$ $(= P_1^1 P_2^2 P_3^3)$. The number of multiplications required in the computation of the conditional probabilities, $P(\mathcal{E}_l|Z)$'s, of the data association hypotheses at different levels in the hypothesis tree is listed in Table-1.

In general, the computation for P_j^t, P_0, $(P_0)^2$, ..., and $(P_0)^{\min(n,m)}$ is called preprocessing, which facilitates the use of common factors in the computation of the conditional probabilities of the data association hypotheses. The details of the multiplications required in the computation of the conditional probabilities of the data association hypotheses at each level of the hypothesis tree is summarized in Table-2.

Let ND_i denote the number of nodes at level i and M be the total number of multiplications required in the computation of $P(\mathcal{E}_l|Z)$'s. From Table-2, it can be inferred that

$$M = ND_1 + 2\sum_{i=2}^{\min(n,m)-1} ND_i + ND_{\min n,m}$$
$$< 2\sum_{i=0}^{\min(n,m)} ND_i. \tag{11}$$

(11) shows that M is less than twice the number of data association hypotheses if preprocessing and common factors are used. This enables us to analyze the computational complexity of the DFS algorithm in terms of the number of data association hypotheses or the number of nodes in the hypothesis tree.

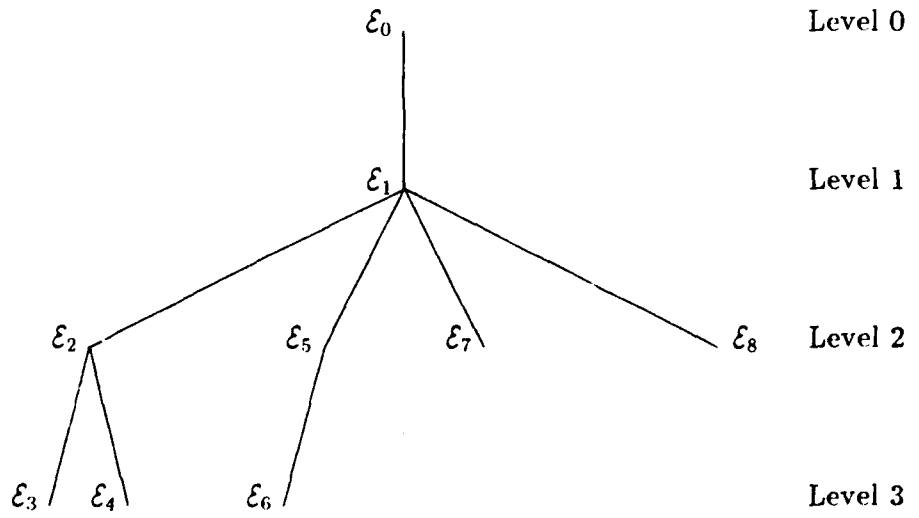


Figure 4 The partial hypothesis tree for the example given in this section

Table 1 Number of multiplications required in the computation of $P(\mathcal{E}_l|Z)$'s

Conditional Probabilities	Number of Multiplications
$P(\mathcal{E}_0|Z)$	0
$P(\mathcal{E}_1|Z)$	1
$P(\mathcal{E}_2|Z)$, $P(\mathcal{E}_5|Z)$, $P(\mathcal{E}_7|Z)$, $P(\mathcal{E}_8|Z)$	2
$P(\mathcal{E}_3|Z)$, $P(\mathcal{E}_4|Z)$, $P(\mathcal{E}_6|Z)$	1

Table 2 Number of multiplications required in the computation of $P(\mathcal{E}_l|Z)$'s in general

Hypotheses at Level i	Number of Multiplications
0	0
1	1
2	2
$\vdots$	2
$\min(n, m) - 1$	2
$\min(n, m)$	1

3.4 Computational Complexity

With the preprocessing discussed in the previous subsection, the computational complexity analysis is simplified as counting the number of data association hypotheses or the number nodes in the hypothesis tree. In general, as we can expect, it is impossible to count the number of the nodes in the hypothesis tree for any situation. In this subsection, the computational complexity is analyzed in two cases, the worst case and the average case. Suppose that there are n targets and m measurements. According to the mathematical model for data association, m variables, X_j $(j = 1, 2, \ldots, m)$, are used to denote the m measurements. Each variable X_j is defined on a set Z_j. In the worst case, Z_j assumes the following form:

$$Z_j = \{0, 1, 2, \ldots, n\} \quad \text{for } j = 1, 2, \ldots, m. \tag{12}$$

In (12), each Z_j contains n non-zero values. In the average case, however, the number of non-zero values in each Z_j is much less than the total number, n, of targets. In the following, the worst case analysis is considered first. Then, the average case analysis is discussed.

3.4.1 The Worst Case Analysis

Let $S(n, m)$ denote the number of data association hypotheses or the number of nodes in the hypothesis tree in the worst case. Suppose, $n = 1$. In this case, either this target is hypothesized to be associated with one of the m measurements or all m measurements are hypothesized to originate from clutter. Therefore, $S(1, m) = m + 1$. Similarly, when $m = 1$, the total number of data association hypotheses, $S(n, 1)$, is equal to $n + 1$. In general, the DFS algorithm starts at the root of a hypothesis tree, where all variables are zero. Then, X_1 is set to 1. The total number of valid combinations of non-zero values of the variables X_j $(j = 2, 3, \ldots, m)$ is $S(n - 1, m - 1)$. Since $X_1 = 1$, the remaining $m - 1$ variables X_j $(j = 2.3, \ldots, m)$ can only have $n - 1$ non-zero values, i.e., $2, 3, \ldots, n$. After

all combinations of non-zero values of the variables X_j $(j = 2, 3, \ldots, m)$ are generated, X_1 is set to 2. As just discussed, the total number of valid combinations of the variables X_j $(j = 2, 3, \ldots, m)$ is $S(n \quad 1, m \quad 1)$. This process can go on until all non-zero values of X_1 are used. Since there are n different non-zero values for variable X_1, the number of data association hypotheses in which X_1 is non-zero is $nS(n-1, m-1)$. Once all the data association hypotheses with $X_1 \neq 0$ are generated, the DFS algorithm goes back to the root of the hypothesis tree and resets X_1 to 0. The remaining data association hypotheses are then generated based on $m - 1$ variables X_j $(j = 2, 3, \ldots, m)$ and n non-zero values for each variable. Therefore, the number of data association hypotheses with $X_1 = 0$ is $S(n, m-1)$. It is obvious now that the number of data association hypotheses or the number nodes in a hypothesis tree in the worst case can be represented by a difference equation,

$$S(n, m) = nS(n-1, m-1) + S(n, m-1). \tag{13}$$

with boundary conditions,

$$\begin{aligned} S(1, m) &= m + 1, \\ S(n, 1) &= n + 1. \end{aligned}$$

Since (13) is not a constant coefficient difference equation, its solution is not very easy to get. To avoid the trouble, return to the structure of the hypothesis tree. Recalling that the nodes at level i represent the hypotheses that i out of m measurements are from i out of n targets. There are C_m^i choices to get any i out of m measurements, where

$$C_m^i = \begin{cases} \frac{m!}{i!(m-i)!} & \text{for } 1 \leq i \leq m \\ 1 & \text{for } i = 0 . \end{cases}$$

Among n targets, there are A_n^i different ways of associating the i measurements with the i out n targets, where

$$A_n^i = \begin{cases} n(n \quad 1) \cdots (n - i + 1) & \text{for } 1 < i \leq n \\ 1 & \text{for } i = 0 . \end{cases}$$

So, the number of nodes in the tree is

$$S(n,m) = \sum_{i=0}^{\min(n,m)} A_n^i C_m^i. \tag{14}$$

It can be shown that the solution satisfies (13). From the solution, it is apparent that the computational cost is highly increasing with the increase of n and m. In Table-3, some $S(n,m)'s$ are listed with different numbers of targets and measurements. From the listed values, we can see that even when the number of targets is as small as six, the number of nodes in the hypothesis tree could be greater than one million in the worst case.

Table 3 Sample values of $S(n,m)$

n	m	$S(n,m)$	n	m	$S(n,m)$
2	4	21	4	8	3,393
3	6	229	6	12	1,442,173

In practical situations, however, when the densities of target and clutter are not very high, the number of non-zero values in each Z_j is much less than that of measurements, m. Consequently, the number of hypotheses is much less than $S(n,m)$. One can derive a tighter upper bound by introducing additional assumptions to modify (13). For example, the number of non-values in each Z_j is less than or equal to two, which means that any measurement cannot be shared by more than two targets. With this assumption, (13) may be changed to

$$S(n,m) = 2S(n-1,m-1) + S(n,m-1). \tag{15}$$

In next subsection, a simple practical upper bound for the number of data association hypotheses is derived in the average case where the number of non-zero values for each variable X_j is much less than that of measurements, m.

3.4.2 The Average Case Analysis

Commonly, it is assumed that there are on the average two measurements inside the validation gate of each target. In other words, the number of non-zero values in each Z_j is less than or equal to two under this assumption. For each target, there are three different associations with the two measurements. That is, the target is hypothesized to be associated with one of the two measurements and the target is not associated with any one of the two measurements. Therefore, on the average, the number of data association hypotheses is bounded by

$$S^*(n,m) = \overbrace{3 * 3 * \cdots * 3}^{n} = 3^n. \tag{16}$$

In the example discussed in Section 2, there are 2 targets and 3 measurements and there are 2 measurements inside the validation gate of each target. The number of data association hypotheses obtained from the worst case analysis is 13 and the upper bound obtained from (16) is 9. The actual number of data association hypotheses in this case is 8. The upper bound given in (16) can be easily extended to the case where the number of measurements inside the validation gate of each target is m^t. That is,

$$S^*(n,m) = \prod_{t=1}^{n}(1 + m^t). \tag{17}$$

4 Direct Computation of β_j^t

As discussed in the previous section, the computational cost for the generation of the data association hypotheses is very high when n, the number of targets in a cluster, and m, the number of measurements, are relatively large. To further speed up the computation of the *a posteriori* probability β_j^t, an algorithm which applies to certain special situation is proposed in this section. This procedure does not requires the generation of the data association hypotheses. This algorithm is applicable when the density of targets is moderate. Specifically, the formulae for computing β_j^t directly are developed for the cases when n equals 1, 2, 3, or 4.

When a cluster is composed of only one target the PDAF is applied. Using the notations in (3) and (4), we have

$$\beta_j^t = P_j^t \quad \text{for } j = 1, 2, \ldots, m, \tag{18}$$

$$\beta_0^t = P_0. \tag{19}$$

Every β_j^t is normalized by

$$c = P_0 + \sum_{j=1}^{m} P_j^t = \sum_{j=0}^{m} \beta_j^t. \tag{20}$$

The normalized β_j^t is the one occurring in the PDAF presented in [1] with $P_G = 1$, where P_G is the probability for the correct measurement to fall inside the validation gate.

For the example given in Section 2, there are 2 closely spaced targets and 3 measurements are received. One of the 3 measurements is inside the intersection of the validation gates of the two targets. Therefore, these two targets are in the same cluster. In this case, each β_j^t is a summation of the conditional probabilities of some data association hypotheses. For example,

$$\beta_1^1 = P(\mathcal{E}(\hat{\Omega}_1)|Z) + P(\mathcal{E}(\hat{\Omega}_2)|Z) + P(\mathcal{E}(\hat{\Omega}_3)|Z).$$

If the notations in (3) and (4) are used, the above equation can be simplified as,

$$\beta_1^1 = P_1^1 P_0 + P_1^1 P_2^2 + P_1^1 P_3^2$$

$$P_1^1(P_0 \cdot P_2^2 \cdot P_3^2). \tag{21}$$

In (21), there is no reference to the conditional probabilities of data association hypotheses $\hat{\Omega}_1$, $\hat{\Omega}_2$, and $\hat{\Omega}_3$. β_1^1 is directly computed from P_j^t and P_0. In general, for 2 targets and m measurements, β_j^t can be obtained by the following equations. For $j = 1, 2, \ldots, m$, $t_i = 1, 2$, for $i = 1, 2$, and $t_1 \neq t_2$,

$$\beta_j^{t_1} \quad P_j^{t_1}(P^{t_2} \cdot P_j^{t_2}) \tag{22}$$

$$\beta_0^{t_1} \quad P_0 P^{t_2} \tag{23}$$

where

$$P^t = P_0 + \sum_{j=1}^{m} P_j^t \quad \text{for } t = 1, 2. \tag{24}$$

In (22), the term $P_j^{t_1} P_j^{t_2}$ is dropped because $P_j^{t_1} P_j^{t_2}$ is the conditional probability for measurement j to originate from target t_1 and t_2, which violates the two restrictions [3] given in Section 2.

Similarly, for 3 targets and m measurements, β_j^t can be computed by the family of equations given below. For $j = 1, 2, \ldots, m$, $t_i = 1, 2, 3$, for $i = 1, 2, 3$, and $t_p \neq t_q$ if $p \neq q$,

$$\beta_j^{t_1} = P_j^{t_1}[(P^{t_2} - P_j^{t_2})(P^{t_3} - P_j^{t_3}) - \sum_{\substack{l=1 \\ l \neq j}}^{m} P_l^{t_2} P_l^{t_3}], \tag{25}$$

$$\beta_0^{t_1} = P_0(P^{t_2} P^{t_3} - \sum_{l=1}^{m} P_l^{t_2} P_l^{t_3}). \tag{26}$$

When the number of targets in a cluster is increased to 4, the set of equations for β_j^t becomes much more complex. For $j = 1, 2, \ldots, m$, $t_i = 1, 2, 3, 4$, for $i = 1, 2, 3, 4$, and $t_p \neq t_q$ if $p \neq q$,

$$\beta_j^{t_1} = P_j^{t_1}[\prod_{i=2}^{4}(P^{t_i} - P_j^{t_i}) - (P^{t_2} - P_j^{t_2}) \sum_{\substack{l=1 \\ l \neq j}}^{m} P_l^{t_3} P_l^{t_4} - (P^{t_3} - P_j^{t_3}) \sum_{\substack{l=1 \\ l \neq j}}^{m} P_l^{t_2} P_l^{t_4} -$$
$$(P^{t_4} - P_j^{t_4}) \sum_{\substack{l=1 \\ l \neq j}}^{m} P_l^{t_2} P_l^{t_3} + 2 \sum_{\substack{l=1 \\ l \neq j}}^{m} P_l^{t_2} P_l^{t_3} P_l^{t_4}] \tag{27}$$

$$\beta_0^{t_1} = P_0\Big(\prod_{i=2}^{4} P^{t_i} \quad P^{t_2}\sum_{l=1}^{m} P_l^{t_3}P_l^{t_4} \quad P^{t_3}\sum_{l=1}^{m} P_l^{t_2}P_l^{t_4} -$$
$$P^{t_4}\sum_{l=1}^{m} P_l^{t_2}P_l^{t_3} - 2\sum_{l=1}^{m} P_l^{t_2}P_l^{t_3}P_l^{t_4}\Big) \tag{28}$$

When the number of targets is larger than 4, the equations for computing β_j^t can be derived with careful consideration. However, the complicated equations themselves may require a lot of memory. In a small system, excessive requirement for memory may not be suitable. Therefore, in this section, the equations for direct computation of β_j^t are derived for the cases in which the number of targets in a cluster is less than or equal to 4. The normalizing constant c in all cases is obtained from

$$c = \sum_{l=0}^{m} \beta_l^t \tag{29}$$

In to order to compare with the DFS algorithm, the number of multiplications required in the computation of β_j^t in the algorithm proposed here is estimated from the above equations in all cases. Since the computational costs for obtaining β_j^t in the two algorithms are the same when there is only one target in a cluster, the following discussion is focused on the cases in which the size of a cluster of targets is larger than 1. In the worst case, in the case of 2 targets and m measurements, one multiplication is required by (22) and one by (23). Therefore, the total number of multiplications required to compute all β_j^t's in this case is $2(m-1)$. If the DFS algorithm is applied, the total number of multiplications can be obtained by using (11). Suppose, $m \geq 2$. Then,

$$\begin{aligned} M &= A_2^1C_m^1 + A_2^2C_m^2 \\ &= m(m+1) \\ &\geq 2(m+1) \end{aligned} \tag{30}$$

where $A_2^iC_m^i$ is the number of nodes at level i in the hypothesis tree as discussed in the complexity analysis in the previous section. Similar comparisons can be made between the DFS algorithm and the direct computation formulae for β_j^t in the case of 3 and 4 targets, as shown in Table-4. In Table-4, DC and DFS stand for direct computation

and depth-first search algorithms, respectively. The results in Table-4 are based on the assumption that m is larger than or equal to n. Generally speaking, the number of multiplications required in the direct computation of β_j^t is less than that required in the DFS algorithm as shown in Table-4. However, this is not true when n and m are equal to 4. Therefore, it is quite possible that the number of multiplications required in the direct computation algorithm could be larger than that required in the DFS algorithm when $n > 4$ and m is almost equal to n. In the implementation, it would be a good idea to combine the two algorithm together. For each cluster of targets, the direct computation algorithm is applied if n, the size of the cluster, is less than or equal to 4. Otherwise, the DFS is applied.

Table 4 Number of multiplications required in the two algorithms

n	DC	DFS
2	$2(m+1)$	$m(m+1)$
3	$3(m^2+2m+2)$	$m(m^2+3m-1)$
4	$4(5m^2+6m+6)$	$m(m^3+2m^2+m+2)$

5 Approximation of β_j^t

Unlike the two algorithms which have been developed in the previous sections to compute β_j^t efficiently, in this section, an algorithm is proposed to compute β_j^t approximately without computing the conditional probabilities of the data association hypotheses in the JPDAF. In the original JPDAF, the interference from all the targets in the same cluster is considered in the computation of β_j^t. This results in high computational cost for the data association. To reduce the computational cost for the data association, only the interference from the neighbors of target t is considered in the approximation of β_j^t. A target is a neighbor of target t if there is at least one measurement inside the intersection of the validation gates of the two targets. It is not necessary that two targets in the same cluster are neighbors. For example, targets 1, 2, and 3 are in the same cluster if targets 1 and 2 are neighbors of target 3. However, if there is no measurement inside the intersection of the validation gates of targets 1 and 2, targets 1 and 2 are not neighbors.

In the direct computation algorithm, each β_j^t is computed as following.

$$\beta_j^t = P_j^t F_j^t \quad \text{for } t = 1, 2, \ldots, n, \text{ and } j = 0, 1, 2, \ldots, m. \tag{31}$$

In (31), F_j^t is the interference from the other targets in the same cluster. Comparing (31) with (18), the interference F_j^t may be considered as a weight on P_j^t. To see what effect of the weight F_j^t has on the value of β_j^t, let us consider the example in Section 2. In that example, there are 2 targets and 3 measurements. Among the 3 measurements, measurement 2 is inside the intersection of the validation gates of the two targets. Using (22) and (23), we have

$$\beta_0^1 = P_0(P_0 + P_2^2 + P_3^2) = P_0 F_0^1, \tag{32}$$

$$\beta_1^1 = P_1^1(P_0 + P_2^2 + P_3^2) = P_1^1 F_1^1, \tag{33}$$

$$\beta_2^1 = P_2^1(P_0 + P_3^2) = P_2^1 F_2^1. \tag{34}$$

From the above equations, it is obvious that F_2^1 is less than both F_0^1 and F_1^1. This means

that more weight is put on measurement 1 than on measurement 2 since measurement 2 is inside the intersection of the validation gates of the two targets. In other words, a measurement is less likely to come from target t if the measurement is inside the intersection of the validation gates of target t and other targets. If F_j^t's are equal for $j = 0, 1, \ldots, m$, then there is either no interference from any other target or the effect of the interferences on the values of β_j^t's are equal. This happens when there is no intersection between the validation gates of target t and the other targets. The value of β_j^t obtained from the joint data association with the other targets is the same as that of β_j^t from the data association for target t alone. Generally, if targets are not grouped into clusters, the value of β_j^t for each target and each measurement is the same as that of β_j^t after targets are grouped into clusters. However, much more computation is needed to compute β_j^t if targets are not grouped into clusters. So, it is very important that targets are grouped into clusters before considering data association.

Even after clustering, the computation to obtain F_j^t could be still very costly if n, the number of targets in a cluster, and m, the number of measurements, are relatively large as discussed in the previous sections. To reduce the computational cost, in the approximation of β_j^t, only the interference from the neighbors of target t is considered in the computation of F_j^t. Let B^t denote the set of neighbors of target t. That is, a target belongs to B^t if there is at least one measurement inside the intersection of the validation gates of this target and target t. Then, the computation of F_j^t is simplified as shown below.

$$F_j^t = P_0 + \sum_{i \in B^t} \sum_{\substack{l=1 \\ l \neq j}}^{m} P_l^i \quad \text{for } j = 1, 2, \ldots, m, \tag{35}$$

$$F_0^t = P_0 + \sum_{i \in B^t} \sum_{l=1}^{m} P_l^i. \tag{36}$$

Substitute F_j^t into (31), to obtain

$$\beta_j^t = P_j^t F_j^t \quad \text{for } t = 1, 2, \ldots, n, \text{ and } j = 0, 1, 2, \ldots, m. \tag{37}$$

Finally, β_j^t is normalized by the constant c in (29).

From the manner in which F_j^t is computed, this algorithm is actually a approximation of the direct computation algorithm presented in the previous section. When $n = 2$, (37) is equivalent to (22) and (23). Comparing with PDAF [2], the computational cost of this approximation algorithm is increased slightly because of computing F_j^t. Furthermore, it is expected that the performance of the approximation algorithm could be close to that of the JPDAF because the interference from neighboring targets of target t is considered in the computation of β_j^t. The computer simulation of the approximation algorithm is presented in the next section.

6 Computer Simulation

In the simulation, the dynamic models for the targets have been digitized using the sampling period T normalized to 1s and the state vectors have been represented in 2-dimensional Cartesian coordinates (or in $X - Y$ plane). Furthermore, only position measurements have been assumed to be available. The surveillance region used in the simulation is a $35km$ by $35km$ square and the initial positions and velocities of 16 targets in 2-dimensional Cartesian coordinates are given in Table-5. The performance of the three algorithms proposed are tested in two separate cases in the simulation. In the first case, suppose that the dynamic characteristics of all targets are known, i.e., every target moves at a constant velocity. In the second case, however, some targets could make a maneuver at any time. In the simulation, target 10 and 11 are capable of making a turn with a centripetal acceleration. The maneuver parameters of the two targets are given in Table-6.

Since every target is nonmaneuvering in the first case, the generic target dynamic model has the following form:

$$\mathrm{x}(k+1) = F\mathrm{x}(k) + G\mathrm{w}(k) \tag{38}$$

$$\mathrm{z}(k) = H\mathrm{x}(k) + \mathrm{v}(k), \tag{39}$$

where

$$\mathrm{x}(k) = \begin{pmatrix} x(k) & \dot{x}(k) & y(k) & \dot{y}(k) \end{pmatrix}', \qquad \mathrm{w}(k) = \begin{pmatrix} w_1(k) & w_2(k) \end{pmatrix}',$$

$$E\{\mathrm{w}(k)\} = 0, \qquad E\{\mathrm{w}(k)\mathrm{w}'(j)\} = Q\delta(k-j),$$

$$F = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad G = \begin{pmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{pmatrix},$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

and

$$E\{\mathrm{v}(k)\} = 0, \qquad E\{\mathrm{v}(k)\mathrm{v}'(j)\} = R\delta(k-j).$$

For maneuvering targets, a higher order of dynamic model is required to describe the characteristics of these targets. In the simulation, the modified Singer's model developed in [12] is used for all targets in the second case. The state and observation equations in the modified Singer's model are described by

$$\mathrm{x}_m(k+1) = F_m\mathrm{x}_m(k) + \mathrm{u}(k) \tag{40}$$

$$\mathrm{z}(k) = H_m\mathrm{x}_m(k) + \mathrm{v}(k). \tag{41}$$

In (40) and (41), $\mathrm{x}_m(k)$, F_m, and H_m are defined as

$$\mathrm{x}_m(k) = \begin{pmatrix} x(k) & \dot{x}(k) & \ddot{x}(k) & y(k) & \dot{y}(k) & \ddot{y}(k) \end{pmatrix}',$$

$$F_m = \begin{pmatrix} f(T) & 0 \\ 0 & f(T) \end{pmatrix}, \qquad f(T) = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix},$$

$$H_m = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

The statistical properties of $\mathrm{v}(k)$ in (41) are identical to those of $\mathrm{v}(k)$ in (39). In (40) and (41), the two noise terms, $\mathrm{u}(k)$ and $\mathrm{v}(k)$ are still assumed to be mutually independent. The state noise $\mathrm{u}(k)$ is a zero-mean Gaussian white random process with covariance matrix $Q_m(k)$ of the following form:

$$Q_m(k) = 2\alpha \begin{pmatrix} \sigma_x(k)q(T) & 0 \\ 0 & \sigma_y(k)q(T) \end{pmatrix}, \qquad q(T) = \begin{pmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{pmatrix},$$

where α is the maneuver correlation coefficient, and $\sigma_x(k)$ and $\sigma_y(k)$ are, respectively, the X and Y components of the maneuver magnitude variance. Both $\sigma_x(k)$ and $\sigma_y(k)$ are adaptively estimated in the simulation.

The correct returns from a target are generated by adding noise to the computed true position of the target. The standard deviation of the measurement noise has been selected as $\sqrt{R_{ii}} = 0.15km$ for both the X and Y components. The correct return would pass a detector with probability of detection $P_D = 0.99$. The clutter is generated uniformly over the whole surveillance region. The total number of clutter returns observed in the region is a Poisson random number. The density of the clutter, λ, is selected to be $0.05/km^2$. This has given an average of 0.5-2 clutter per gate in the examples given below. The threshold g^2 for the validation gate is set to 17.0.

The efficiency of the three algorithms is estimated in terms of CPU time used in the data association in a single sample run. The simulation is performed on a VAX 8550. The exact CPU time for each example could vary from time to time, depending upon the load of the computer. The robustness of the three algorithm is evaluated using the success rate in tracking both maneuvering and nonmaneuvering targets in clutter. The success rate means the number of sample runs finished without lost tracking of a single target in 100 different sample runs.

In the simulation, three examples are used to track 5, 9, and 16 targets with different dynamic models given in (38) and (40). The initial parameters of the 5 targets in the first example are taken from targets 4, 6, 7, 11, and 13 and the initial parameters of the 9 targets in the second example are taken from targets 1, 2, 4, 6, 7, 8, 10, 11, and 16 in Table-5. In the third example, all targets listed in Table-5 are used. When all targets are supposed to be nonmaneuvering, the three examples are shown in Figs. 5, 7, and 9. If the modified Singer's model in (40) is used for all targets, the trajectories of all targets in the three examples are shown in Figs. 6, 8, and 10.

The CPU times used in the data association in one sample run in the three algo-

rithms are listed in Table-7. In Table-7, DFS, AC, and DC stand for depth-first search, approximate computation, and direct computation algorithms, respectively. In the first example, as shown in Figs. 5 and 6, the largest size of cluster of targets is 3. But most of the time during simulation, the sizes of clusters are 1 and, occasionally, 2. When there is one target in a cluster, a PDAF is applied for tracking this target. In the three algorithms, the equations used to compute β_j^t are the same when the size of a cluster is 1. As a result, in this example, the CPU times used in the data association are almost the same in the three algorithms for tracking nonmaneuvering and maneuvering targets. In the second example, as shown in Figs. 7 and 8, the total number of targets is increased to 9 and the largest size of cluster of targets is increased to 4. Most of the time in the simulation, the size of the largest cluster at each time instant is about 2. Therefore, the increases of the CPU times in the three algorithms are roughly proportional to the increase of the number of targets. However, in the third example as shown in Figs. 9 and 10, the size of the largest cluster is frequently ranging from 6 to 8, and it is sometimes as large as 9. This leads to a significant increase of the CPU time used for data association in the DFS algorithm. For the approximate computation (AC) algorithm, the increase of the CPU time is proportional to the increase of the number of targets because of the way in which the data association is considered in this algorithm. Although the AC algorithm is much more efficient than the DFS algorithm when the targets are nonmaneuvering, the AC algorithm is not suitable for use to track 16 targets when there are possible maneuvers. Since the size of the cluster of targets is frequently larger than 4 in this example, the direct computation (DC) algorithm is not applicable.

The success rates of the three algorithms are listed in Table-8. When targets are nonmaneuvering, target 11, as shown in Fig. 9, causes most of the mistracking. However, when the dynamic model given in (40) is used, the two maneuvering targets are the major cause of the mistracking. Of course, the different layout of the targets could lead to different success rates. Especially, the success rates might be improved if either target 11 is removed or it is moved to a different position. However, the current layout of the

targets could help us understand the limitation of the joint data probabilistic association technique. In all examples, the available success rates of the DFS and the DC algorithms are the same, as expected, when the density of targets is not high. But the DC algorithm is computationally more efficient than the DFS algorithm. The high success rate of the DFS algorithm is accompanied by high computational cost in comparison with the AC algorithm. The success rate of the AC algorithm in the case of tracking 16 maneuvering targets is not available in Table-8 due to the same reason given above. The performance of the DFS and the AC algorithms deteriorates with the increase in the number of targets. This difficulty is caused by the increase of the number of measurements inside each gate. Therefore, additional information is needed to reduce the uncertainty in the data association. Practically, the choice of the three algorithm in the design of a tracking system will be dependent upon the computational capability and the requirement of tracking accuracy.

Table 5 The initial positions and velocities of 16 targets

Target	Position (km)		Velocity (km/s)	
i	x	y	$\dot{x}$	$\dot{y}$
1	3.0	3.0	0.72	0.12
2	3.0	7.8	0.72	-0.12
3	10.0	2.0	0.45	0.60
4	6.0	10.0	0.62	0.35
5	6.0	12.0	0.62	0.35
6	5.0	24.0	0.50	0.22
7	5.0	30.0	0.50	-0.50
8	4.0	34.0	0.70	-0.31
9	5.0	20.5	0.62	0.00
10	4.0	18.0	0.70	0.00
11	11.0	7.0	0.45	0.55
12	18.0	34.2	0.10	-0.68
13	3.0	29.0	0.71	0.00
14	11.0	33.0	0.60	-0.60
15	12.0	7.0	0.35	0.65
16	4.0	26.0	0.60	0.25

Table 6 The maneuver parameters of target-10 and target-11

Trajectory i	Acceleration a_c (m/s^2)	Turn Started at (s)	Turn Finished at (s)
10	-0.05	10	25
11	0.08	15	30

Table 7 CPU Time Used in Data Association in Seconds (one sample run)

# of targets	Nonmaneuvering (38)			Maneuvering (40)		
n	DFS	AC	DC	DFS	AC	DC
5	0.14	0.14	0.09	0.22	0.11	0.12
9	0.36	0.17	0.24	0.41	0.19	0.24
16	35.98	0.39	N/A	78.80	N/A	N/A

Table 8 The success rates of the three algorithms in 100 runs

# of targets	Nonmaneuvering (38)			Maneuvering (40)		
n	DFS	AC	DC	DFS	AC	DC
5	100	100	100	94	94	94
9	100	99	100	70	69	70
16	85	80	N/A	32	N/A	N/A

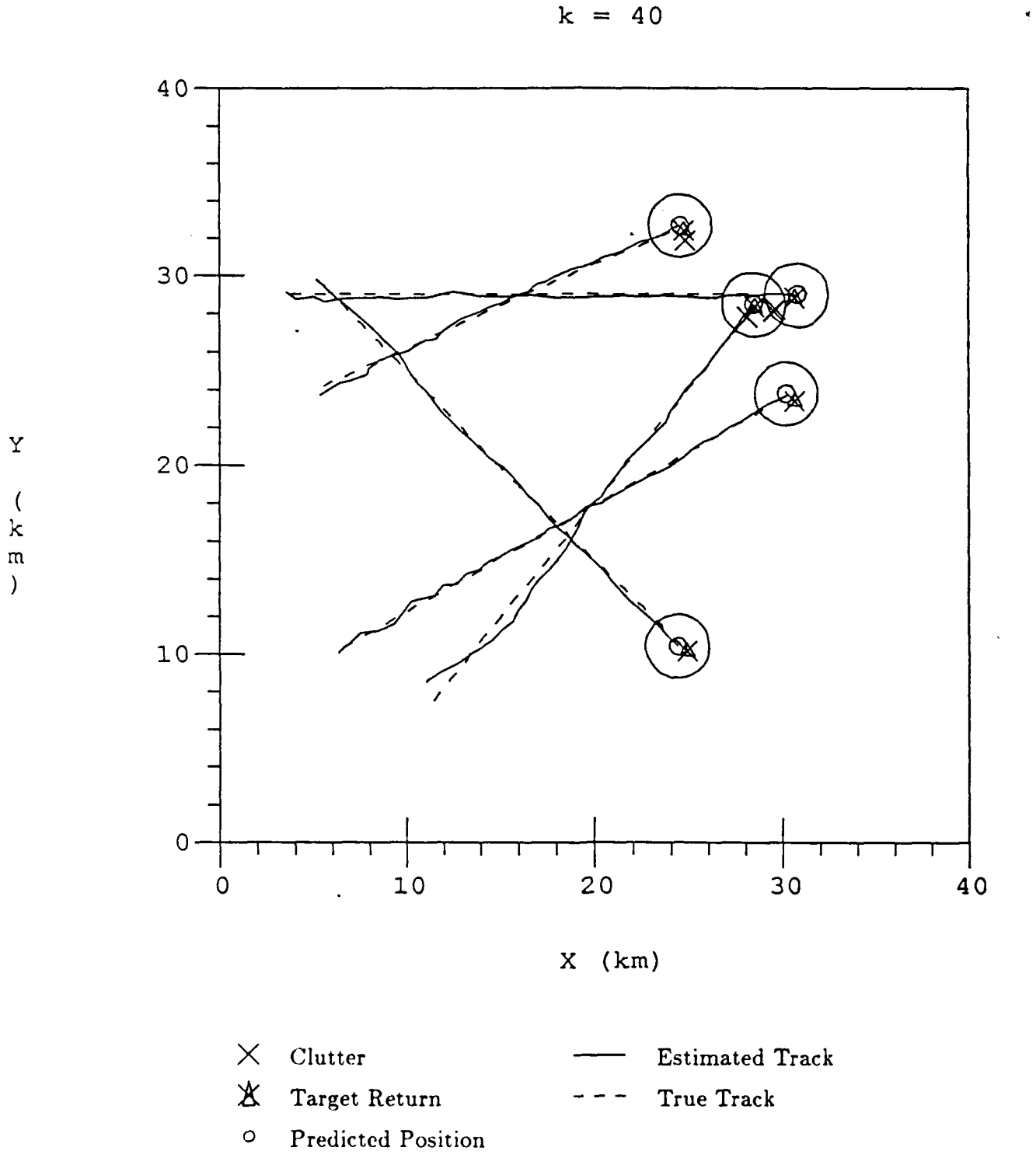


Figure 5 Tracking 5 nonmaneuvering targets.

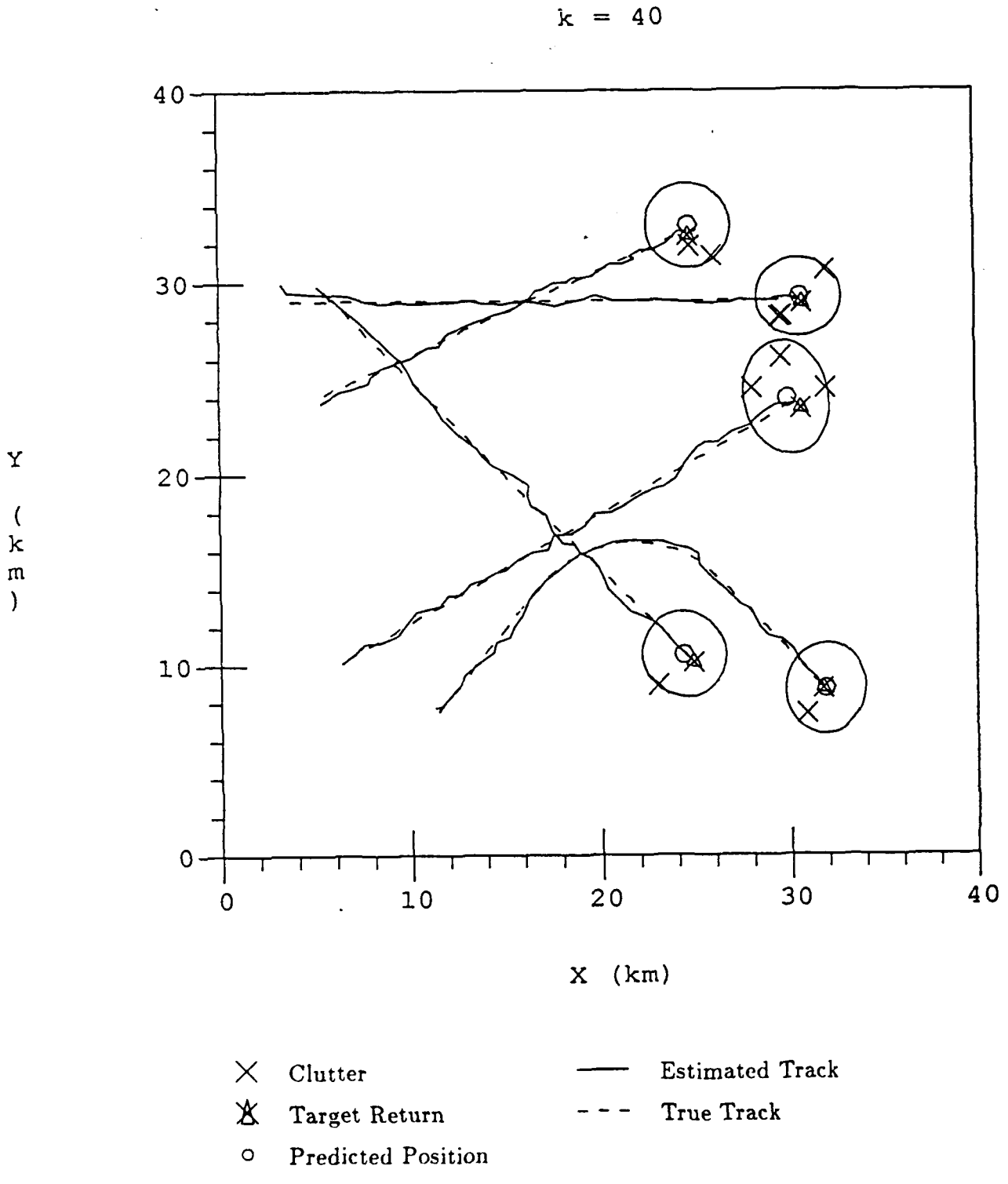


Figure 6 Tracking 5 maneuvering targets.

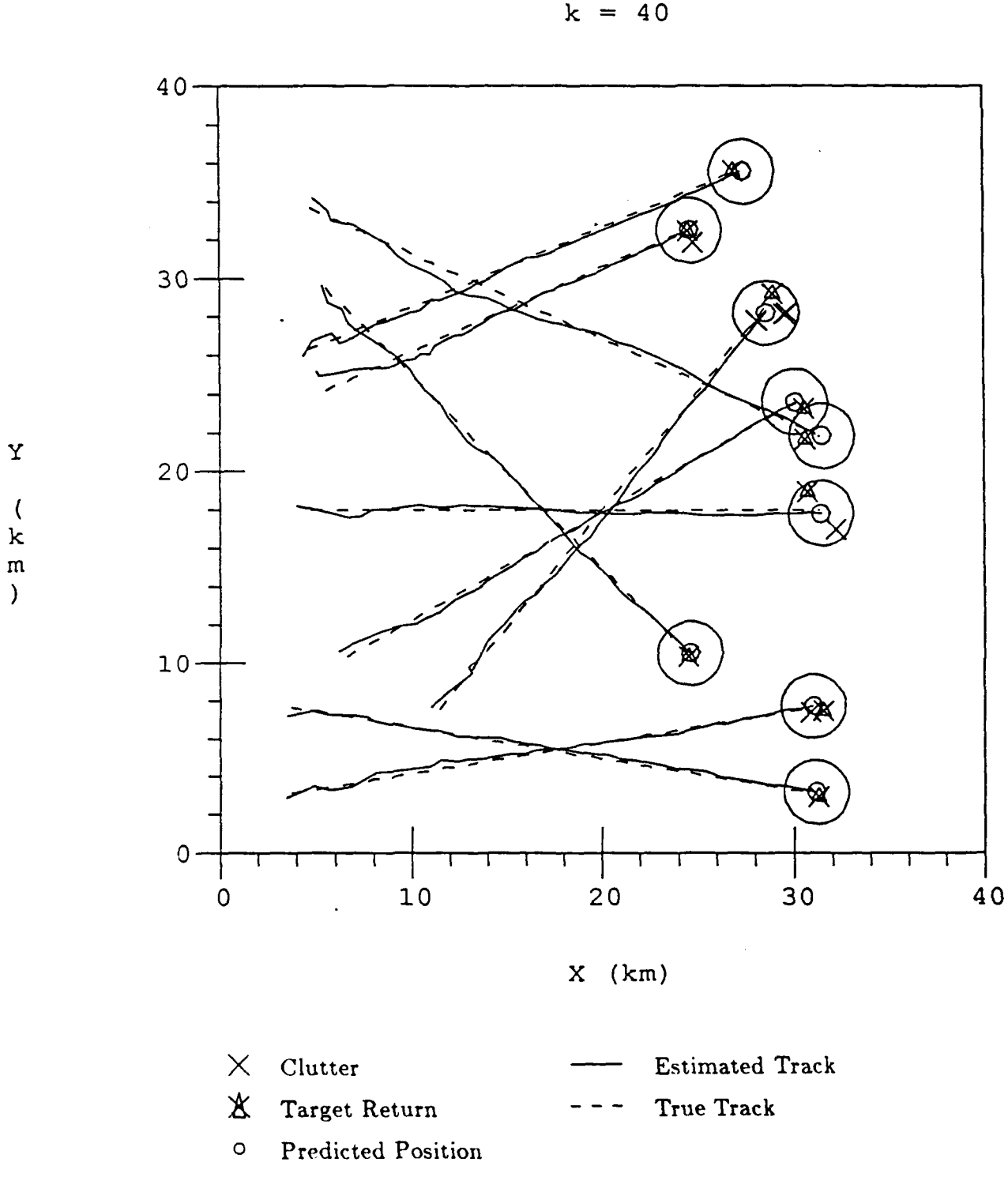


Figure 7 Tracking 9 nonmaneuvering targets.

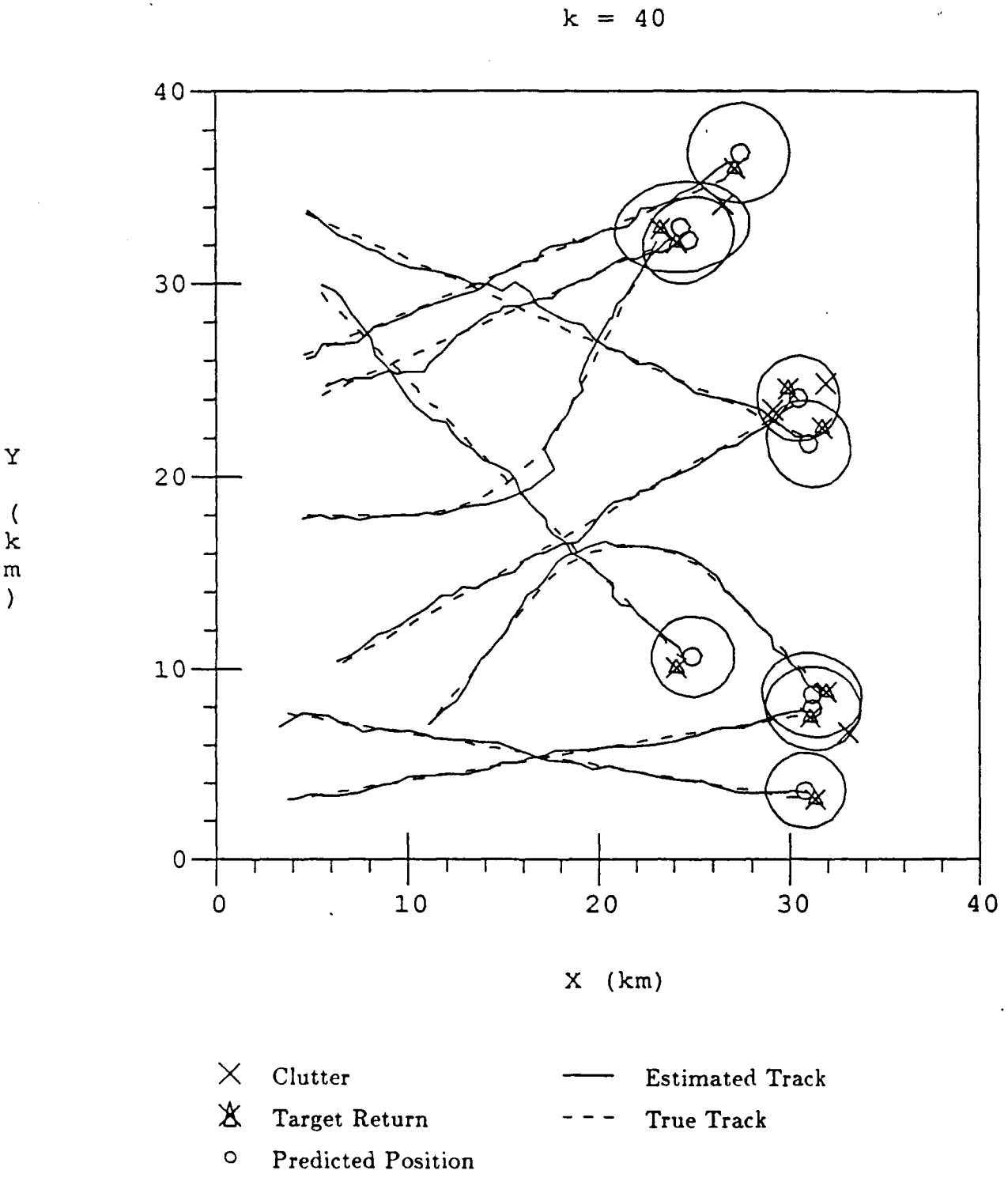


Figure 8 Tracking 9 maneuvering targets.

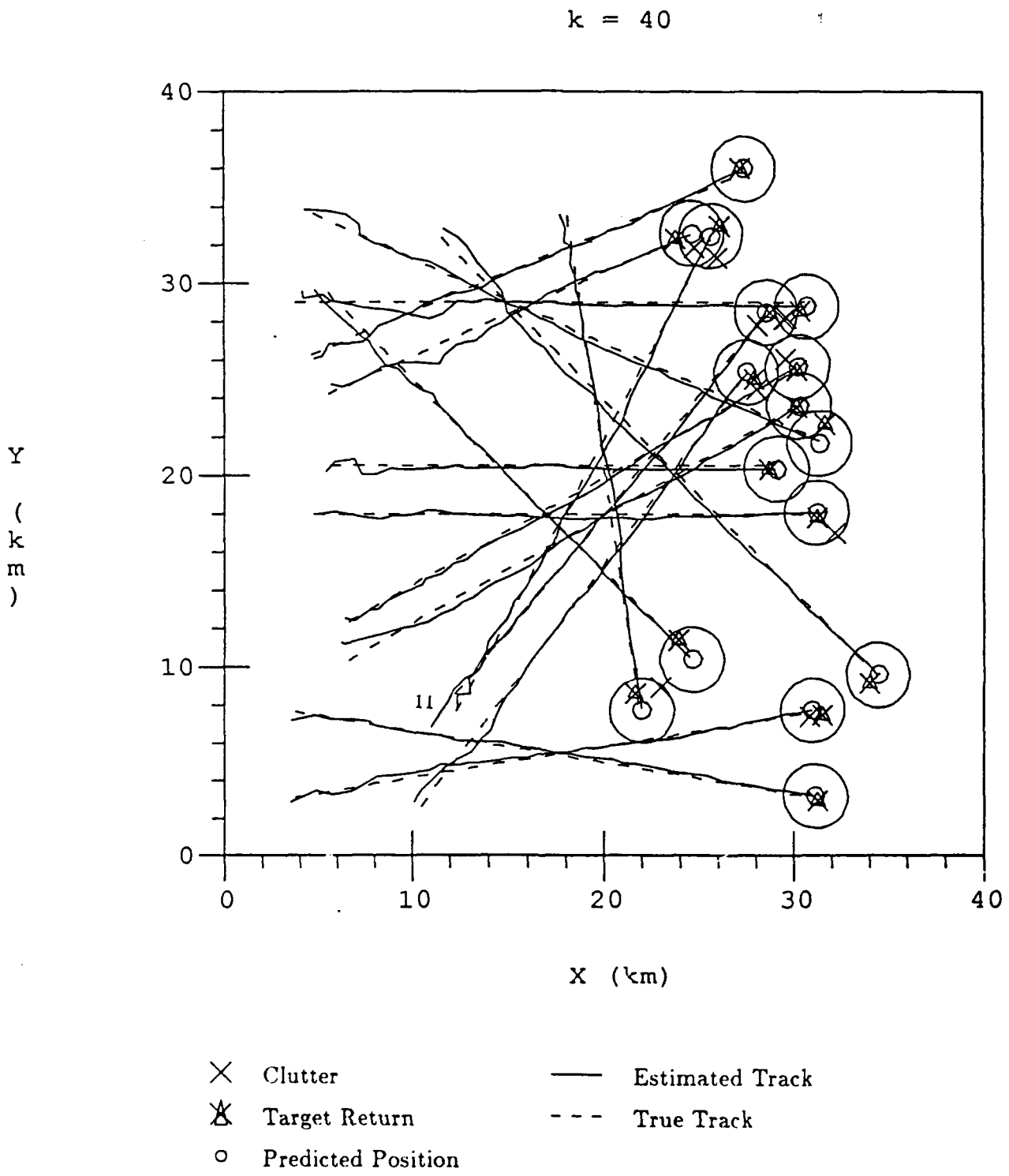


Figure 9 Tracking 16 nonmaneuvering targets.

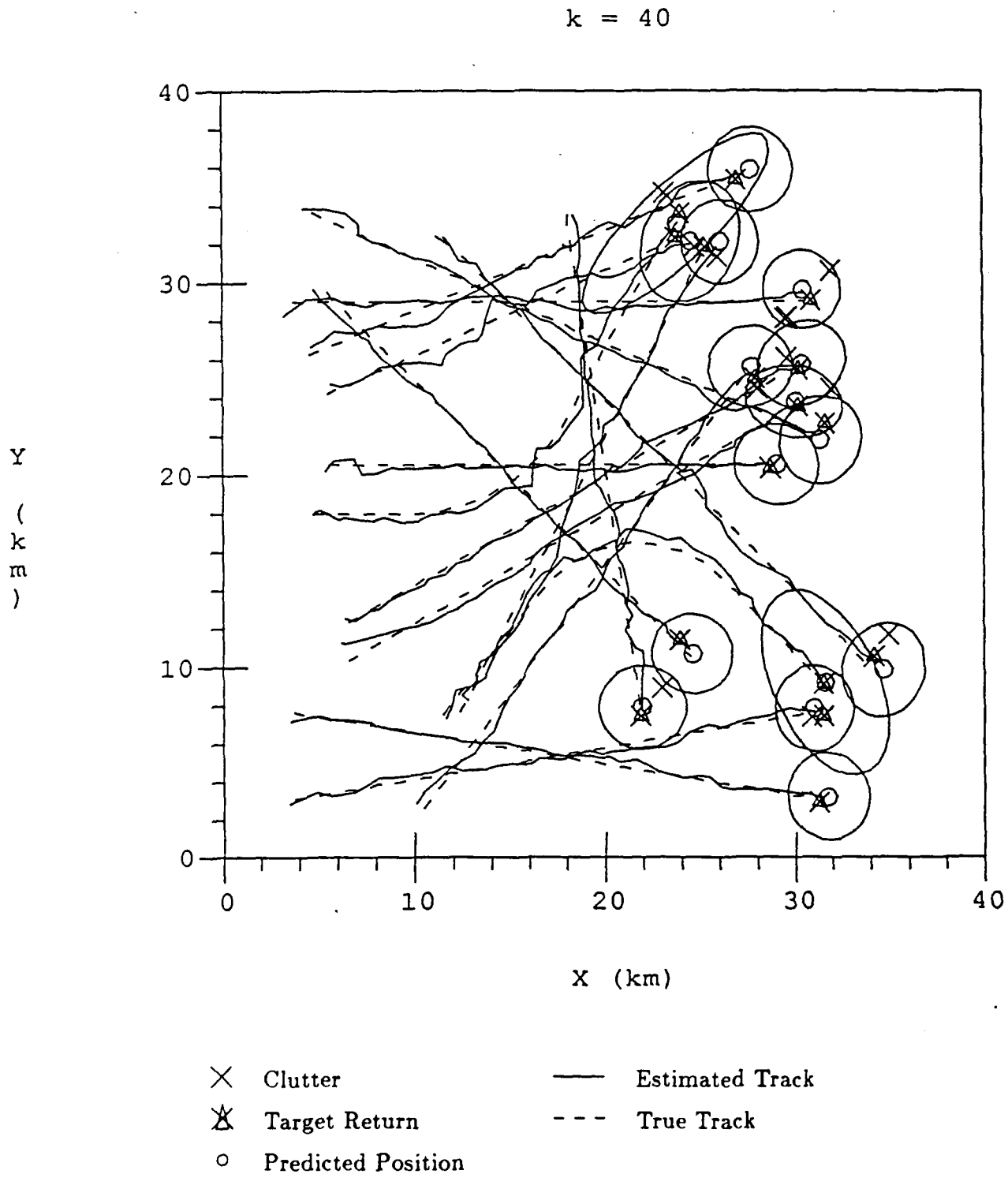


Figure 10 Tracking 16 maneuvering targets.

7 Conclusion

In this paper, three algorithms have been proposed for fast computation of the *a posteriori* probability β_j^t in the JPDAF. As shown in the simulation, the CPU time is increased drastically when the largest size of the cluster of targets is relatively large. Therefore, the DFS algorithm is suitable for implementation in a tracking system and in a ground-based surveillance system with a large centralized computational capability. Furthermore, the JPDAF implemented with the DFS algorithm could serve as a standard for any future attempt to approximate the JPDAF. The advantages of the AC algorithm proposed in this paper are that the computation of the *a posteriori* probability β_j^t is very efficient and that the success rate is relatively high in comparison with the JPDAF without any approximation. Especially, the AC algorithm is suitable for implementation in a multiprocessor system. The DC algorithm is more efficient than the DFS algorithm and can be implemented in a multiprocessor system. However, right now, it is only applicable when the density of targets is not high.

Although the discussion here is focused on the target-oriented approach, i.e., JPDAF [3], it is not difficult to extend the DFS algorithm to the measurement-oriented approach [4]. Furthermore, the DFS-based technique is applicable to three dimensional validation matrices which occur in Markov models of system parameters for maneuvering targets in clutter [10] and in multiscan correlation [4].

References

[1] Y. Bar-Shalom and Thomas E. Fortmann, *Tracking and Data Association*, Academic Press, Inc., 1988.

[2] Y. Bar-Shalom and Edison Tse, "Tracking in a Cluttered Environment With Probabilistic Data Association," *Automatica, Vol. 11* , pp. 451-460, September, 1975.

[3] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association," *IEEE J. Oceanic Engineering,* VOL. OE-8, pp. 173-184, July 1983.

[4] D. B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Trans. Automatic Control,* Vol. AC-24, pp. 843-854, December 1979.

[5] A. K. Mahalanabis, B. Zhou and N. K. Bose, "Improved Multi-Target Tracking in Clutter by PDA Smoothing," *IEEE Trans. Aerospace Electronic System,* Vol. 26, pp. 113-121, January 1990.

[6] R. J. Fitzgerald, "Development of Practical PDA Logic for Multitarget Tracking by Microprocessor," *Proc. American Controls Conference,* pp. 889-898, June 18-20, 1986, Seattle, WA.

[7] R. J. Fitzgerald, "Development of Practical PDA Logic for Multitarget Tracking by Microprocessor," Chapter 1 in *Multitarget-Multisensors Tracking: Advanced Applications* Edited by Y. Bar-Shalom, Artech House Inc. 1990.

[8] D. Sengupta and R. A. Iltis, "Neural Solution to Multitarget Tracking Data Association Problem," *IEEE Trans. Aerospace and Electronic System,* Vol .AES-25, pp. 96-108, January 1989.

[9] V. Nagarajan, M. R. Chidambara and R. N. Sharma, "Combinatorial Problems in Multitarget Tracking — A Comprehensive Solution," *IEE Proc. Part. F Communication Radar and Signal processing.*, Vol. 134, pp. 113-118, February 1987.

[10] D. Sengupta and R. A. Iltis, "Tracking of Multiple Maneuvering Targets in Clutter by Joint Probabilistic Data and Maneuver Association," *Proc. American Control Conference*, pp. 2696-2701, June 21-23, 1989, Pittsburgh, PA.

[11] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms, Theory and Practice*, Englewood Cliffs, N. J., Prentice-Hall, 1977.

[12] N. K. Bose and B. Zhou, "Application of smoothing techniques for tracking maneuvering targets," Technical Report sponsored by SDIO/IST and managed by the Office of Naval Research under contract N00014-86-0542, April, 1989.

7. Reports Distribution List

Scientific Officer Code: 1114SE
Rabinder N. Madan
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Administrative Grants Officer
Office of Naval Research
Resident Representative N66005
Administrative Contracting Officer
The Ohio State University Research Center
1314 Kinnear Road
Columbus, OH 43212-1194

Director, Naval Research Laboratory
Attn: Code 2627
Washington, DC 20375

Defense Technical Information Center
Building 5, Cameron Station
Alexandria, VA 22304-6145

END